NEC

User's Manual

V55PI™

16-bit Microprocessor

Hardware

μPD70433

Document No. U10514EJ5V0UM00 (5th edition)
Date Published June 1997 N

© NEC Corporation 1995
Printed in Japan

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

V20, V30, V20HL, V30HL, V33A, V25, V35, V25+, V40, V50, V40HL, V50HL, V53A, V55PI, and V Series are trademarks of NEC Corporation.

The application circuits and their parameters are for reference only and are not intended for use in actual design-ins.

The information in this document is subject to change without notice.

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated “quality assurance program“ for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

- Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
- Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
- Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

Anti-radioactive design is not implemented in this product.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 253-8311
Fax: 250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Sao Paulo-SP, Brasil
Tel: 011-889-1680
Fax: 011-889-1689

Major Revisions in This Edition

Page	Description
p.4, 5	**1.3 Ordering Information:** μPD70433GD-5BB has been deleted.
p.35	**2.4 (1) Ports:** Recommended connection method of P60 to P63/ ANI0 to ANI3 has been changed.
p.106	**4.8 Bus Hold Function:** Caution has been added.
p.207	**Figure 6-5. Demand Release Mode Transfer Timing:** Data timing has been changed.
p.387	**Figure 12-1. Example of Connecting RESET Pin** has been added.
p.419	**Table 15-1. HALT/STOP Mode Operation States:** "Port" has been added, and "$\overline{\text{IORD}}$, $\overline{\text{IOWR}}$" has been changed.
p.480	**18.6.3 MH encode instruction:** Caution has been added.
p.482	**18.6.4 MR encode instruction:** Caution has been added.
p.539	**B.2 (7) Cautions when using single step mode** has been added.
p.541	**B.2 (8) Cautions when using intelligent DMA mode-1** has been added.

The mark ★ shows major revised points.

PREFACE

Target Readers

This manual is intended for engineers who wish to understand the functions of the μPD70433 (also referred to as V55PI) and design their application systems.

Purpose

This manual is intended for understanding various hardware functions of the μPD70433 described in the following organization.

Organization

The μPD70433 user's manual consists of two volumes: hardware (this manual) and instruction.

Hardware volume	Instruction volume
• General description	• General description
• Pin functions	• Description of instructions
• CPU functions	• Instructions additional to V20/V30 or V25/V35
• On-chip peripheral hardware	• Queue manipulation instruction
• Standby functions	• Codec instruction
• Reset functions	

How to Read This Manual

This manual assumes the reader has general knowledge of electronics, logic circuits, and microcomputers.
In this manual, the product name "μPD70433" is referred to as the V55PI.

For a user who has experience with the μPD70108/70116 (V20™, V30™)
→ the V55PI has compatibility with the V20 and V30 native mode instructions. See **CHAPTER 3 INSTRUCTIONS ADDITIONAL TO V20/V30 OR V25/V35** in the separate **V55PI USER'S MANUAL – INSTRUCTION**.

For a user who has experience with the μPD70320/70330 (V25™, V35™)
→ confirm the V55PI-specific on-chip peripheral hardware referring the section **1.2 Comparison of V55PI and V25 Functions**.
For the differences in software, see **CHAPTER 3 INSTRUCTIONS ADDITIONAL TO V20/V30 OR V25/V35** in the separate **V55PI USER'S MANUAL – INSTRUCTION**.

To understand details of the instruction functions,
→ see the separate **V55PI USER'S MANUAL – INSTRUCTION**.

If you want to have a general understanding of V55PI hardware functions:
→ read this manual from CHAPTER 1 sequentially.

For information on the electrical specifications
→ see the separate **V55PI DATA SHEET**.

For application examples of each function
→ see the separate **APPLICATION NOTE**.

Legend

Data representation weight	:	High digits on the left and low digits on the right
Active-low representations	:	$\overline{\times\times\times}$ (line over the pin and signal names)
Memory map address	:	Low address in the upper section and high address in the lower section
Note	:	Description of **Note** in the text
Caution	:	Information requiring particular attention
Remark	:	Additional explanatory material
Numeral representations	:	Binary ... ×××× or ××××B Decimal ... ×××× Hexadecimal ... ××××H

Prefix of the power of 2 (address space, memory capacity):
K : $2^{10} = 1024$
M : $2^{20} = 1024^2$

Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

- Pamphlet (U10244J)**Note**
- Data Sheet (U11775E)
- User's Manual Hardware (this manual)
- User's Manual Instructions (U10231E)
- Application Note Software (IEA-743)**Note**
- Application Note Facsimile (IEA-727)**Note**
- Application Note Hardware (IEA-750)**Note**
- Special Function Register Table (IEM-5592)**Note**

Documents Related to Development Tools

Document			Document No.
Hardware	IE-70433-BX User's Manual		U10608E
Software	RA70116-I, SP70116-I		
	User's Manual	Assembly Language	EEU-861**Note**
		Operation	EEU-869**Note**

Note The document numbers are those of Japanese-version documents.

CONTENTS

CHAPTER 1 GENERAL DESCRIPTION 1

1.1 Features 2

1.2 Comparison of V55PI and V25 Functions 3

1.3 Ordering Information 4

1.4 Pin Configuration (Top View) 5

1.5 Internal Block 8

1.5.1 Internal block diagram 8

1.5.2 Internal block configuration 9

CHAPTER 2 PIN FUNCTIONS 11

2.1 List of Pin Functions 11

2.1.1 Port functions 11

2.1.2 Non-port functions 13

2.1.3 Pin statuses 16

2.2 Description of Pin Functions 19

2.2.1 P00 to P07 (Port 0) 19

2.2.2 P10 to P16 (Port 1) 19

2.2.3 P20 to P25 (Port 2) 20

2.2.4 P30 to P36 (Port 3) 21

2.2.5 P40 to P47 (Port 4) 23

2.2.6 P50 to P52 (Port 5) 24

2.2.7 P60 to P63 (Port 6) 26

2.2.8 P70 to P77 (Port 7) 26

2.2.9 P80, P81 (Port 8) 27

2.2.10 GND (Ground) 27

2.2.11 V_{DD} (Power Supply) 28

2.2.12 AV_{SS} (Analog V_{SS}) 28

2.2.13 AV_{DD} (Analog V_{DD}) 28

2.2.14 AV_{REF} (Analog Reference Voltage) 28

2.2.15 $\overline{RESET}$ (Reset) 28

2.2.16 X1, X2 (Crystal) 28

2.2.17 CLKOUT (Clock Out) 29

2.2.18 $\overline{ASTB}$ (Address Strobe) 29

2.2.19 $\overline{RD}$ (Memory Read Strobe) 29

2.2.20 $\overline{WRL}$ (Memory Write Low Strobe) 29

2.2.21 $\overline{WRH}$ (Memory Write High Strobe) 29

2.2.22 READY (Ready) 30

2.2.23 $\overline{DEX}$ (Data Exchange) 30

2.2.24 $\overline{RAS}$ (Row Address Strobe) 30

2.2.25 D8/$\overline{D16}$ (Data 8/16) 31

2.2.26 $\overline{BUSLOCK}$ (Buslock) 31

2.2.27 $\overline{POLL}$ (Poll) 31

2.2.28 HLDRQ (Hold Request) 31

2.2.29 $\overline{HLDAK}$ (Hold Acknowledge) 31

2.2.30 AD0 to AD15 (Address/Data 0 to 15) 31
2.2.31 A16 to A23 (Address 16 to 23) 31
2.2.32 $\overline{\text{IORD}}$ (I/O Read) 31
2.2.33 $\overline{\text{IOWR}}$ (I/O Write) 32
2.2.34 $\overline{\text{WDTOUT}}$ (Watchdog Timer Output) 32
2.2.35 $\overline{\text{DMAAK0}}$, $\overline{\text{DMAAK1}}$ (DMA Acknowledge 0, 1) 32
2.2.36 $\overline{\text{TCE0}}$, $\overline{\text{TCE1}}$ (Terminal Count End 0, 1) 32
2.2.37 $\overline{\text{REFRQ}}$ (Refresh Request) 32
2.3 Pin Input/Output Circuits 33
2.3.1 Input/output circuit types 33
2.3.2 Input/output circuits 34
2.4 Recommended Connections of Unused Pins 35

CHAPTER 3 CPU FUNCTIONS 37
3.1 Features 37
3.2 Registers 38
3.2.1 Register banks 38
3.2.2 General registers (AW, BW, CW, and DW) 40
3.2.3 Pointers (SP and BP) and index registers (IX and IY) 40
3.2.4 Segment registers (PS, SS, DS0, and DS1) 41
3.2.5 Expansion segment registers (DS2, DS3) 42
3.2.6 Special-function registers (SFR) 44
3.3 Program Counter (PC) 45
3.4 Program Status Word (PSW) 45
3.4.1 CY (Carry Flag) 46
3.4.2 P (Parity Flag) 47
3.4.3 AC (Auxiliary Flag) 47
3.4.4 Z (Zero Flag) 47
3.4.5 S (Sign Flag) 47
3.4.6 V (Overflow Flag) 48
3.4.7 $\overline{\text{IBRK}}$ (I/O Break Flag) 49
3.4.8 BRK (Break Flag) 49
3.4.9 IE (Interrupt Enable Flag) 49
3.4.10 DIR (Direction Flag) 49
3.4.11 RB0 to RB3 (Register Bank 0 to 3 Flag) 49
3.5 Memory Space 50
3.5.1 Basic memory space 50
3.5.2 Expansion memory space 51
3.5.3 Special-function register area 53
3.5.4 Vector table area 62
3.6 Register File Space 64
3.7 I/O Space 66
3.8 Logical Address and Physical Address 67
3.8.1 Segment method 67
3.8.2 Segment configuration 69
3.8.3 Dynamic relocation 73
3.9 Effective Address 74
3.10 Addressing Mode 75
3.10.1 Instruction address 75

3.10.2 Memory operand address 78

CHAPTER 4 BUS CONTROL FUNCTION 85
4.1 External Bus Control Pins 85
4.2 External Data Bus Width Selection Function 85
4.3 Basic Bus Cycle 85
4.4 Memory Block Control Function 90
4.4.1 Memory block control registers 90
4.5 Wait Function 94
4.5.1 Address wait and data wait 94
4.5.2 Programmable wait function 94
4.5.3 Wait control with READY pin 95
4.5.4 DMA transfer cycle wait 98
4.5.5 Wait in SFR access 98
4.6 $\overline{\text{RAS}}$ Pin Function 99
4.7 Refresh Function 101
4.7.1 Refresh mode register (RFM) 101
4.7.2 Refresh cycle wait control 103
4.7.3 Refresh address 103
4.7.4 Refresh timing 103
4.7.5 Pseudo-SRAM connection 105
4.8 Bus Hold Function 106
4.9 Bus Mastership 107
4.10 Bus Timings 107

CHAPTER 5 INTERRUPT FUNCTIONS 121
5.1 Features 121
5.2 Interrupt Sources 122
5.2.1 Non-maskable interrupt 125
5.2.2 Maskable interrupts 125
5.3 Priority Control 128
5.3.1 Priority control for multiple interrupts 128
5.3.2 Priority control for simultaneously generated interrupts 129
5.3.3 Multiple interrupt control mode selection 130
5.4 Interrupt Response Methods 131
5.4.1 Vectored interrupts 132
5.4.2 Register bank switching 133
5.4.3 Macro service function 138
5.5 Control Registers 172
5.5.1 Interrupt mode control register (IMC) 172
5.5.2 Interrupt mask flag registers (MK0L, MK0H, MK1L, MK1H) 174
5.5.3 Interrupt request control registers (IC9 to IC14, IC16 to IC32, IC36 and IC37) 175
5.5.4 In-service priority register (ISPR) 178
5.6 External Interrupts 179
5.6.1 External interrupt mode registers (INTM0, INTM1) 179
5.7 Software Interrupts 180
5.7.1 General software interrupts 180
5.7.2 Input/output instruction interrupts 181
5.7.3 FPO instruction interrupt 182

5.8 Exception Trap 183
5.9 Times at Which Hardware Interrupt Requests are not Acknowledged 183
5.10 Interrupt Servicing during Block Processing Instruction Execution 184
5.11 Interrupt Servicing during Codec Instruction Execution 184
5.12 Examples of Interrupt Request Service Operations 186
5.13 Interrupt Response Time 194
5.14 Caution 197
5.14.1 When performing SFR access by macro service 197
5.14.2 Interrupt mode control register manipulation 197
5.14.3 Interrupt request generation after macro service ends 197
5.14.4 If an interrupt is not generated 197
5.14.5 Cautions regarding FINT instruction 198

CHAPTER 6 DMA FUNCTION (DMA CONTROLLER) 199
6.1 Features 199
6.2 Basic Operation 200
6.3 Configuration 203
6.3.1 Channel configuration 203
6.3.2 Transfer target 203
6.3.3 Start source 204
6.3.4 Transfer modes 207
6.3.5 Operating modes 209
6.4 Operation 230
6.4.1 Start 230
6.4.2 Stop 231
6.4.3 Suspend 231
6.5 Priority 232
6.5.1 Inter-channel priority 232
6.5.2 Priority with respect to refresh cycle 232
6.5.3 Priority with respect to other control operations 232
6.6 Control Registers 233
6.6.1 DMA mode registers (DMAM0, DMAM1) 233
6.6.2 DMA control registers (DMAC0, DMAC1) 237
6.6.3 DMA status register (DMAS) 240
6.7 DMA Service Channels 242
6.7.1 Terminal counters (TC0, TC1) 242
6.7.2 Terminal counter modulo registers (TCM0, TCM1) 243
6.7.3 DMA up/down counters (UDC0, UDC1) 243
6.7.4 DMA read/write pointers (DPTC0, DPTC1) 243
6.7.5 DMA memory address registers (MAR0, MAR1) 243
6.7.6 DMA compare registers (DCM0, DCM1) 244
6.8 Interrupt Requests 245
6.9 Response Time to DMA Request 246
6.10 DMA Transfer Timing 247
6.11 Caution 260
6.11.1 Restrictions on transfer object specification with main channel DMA controller 260
6.11.2 Restrictions on use of 2-channel operating mode 262
6.11.3 Transfer mode in refresh cycle automatic insertion enable 263
6.11.4 UART receive interrupt request used as a DMA transfer request source 263

CHAPTER 7 SERIAL INTERFACE FUNCTIONS 265
7.1 Features 265
7.2 Communication Protocols 265
7.2.1 UART 266
7.2.2 CSI 266
7.2.3 Control Register 267
7.3 Baud Rate Generator 269
7.3.1 Configuration 269
7.3.2 Operation 270
7.3.3 Control registers 272
7.4 UART Operation 274
7.4.1 Configuration 274
7.4.2 Transmit operation 275
7.4.3 Receive operation 276
7.4.4 Wake-up function 277
7.4.5 Control registers 279
7.4.6 Interrupt requests 287
7.4.7 DMA request 289
7.4.8 Caution 290
7.5 CSI Operation 292
7.5.1 Configuration 292
7.5.2 Operation in 3-wire serial I/O mode (IOE mode) 296
7.5.3 Serial bus interface mode (SBI mode) operation 299
7.5.4 Control registers 305
7.5.5 Interrupt requests 316
7.5.6 Caution 316

CHAPTER 8 PARALLEL INTERFACE FUNCTIONS 317
8.1 Features 317
8.2 Configuration 317
8.3 Operation 320
8.3.1 Input mode 320
8.3.2 Output mode 321
8.4 Control Registers 323
8.4.1 Parallel interface control register 0 (PAC0) 323
8.4.2 Parallel interface control register 1 (PAC1) 328
8.4.3 Parallel interface status register (PAS) 330
8.4.4 Parallel interface acknowledge interval register (PAI1, PAI2) 332
8.4.5 Parallel interface buffer register (PAB) 333
8.5 Interrupt Requests 333
8.5.1 In input mode 333
8.5.2 In output mode 333
8.6 Operation Timing 334

CHAPTER 9 TIMER FUNCTIONS 347
9.1 Features 347
9.2 Configuration 347
9.3 Operation 350

9.3.1 Count function 350
9.3.2 Capture function 354
9.3.3 Compare function 355
9.3.4 Timer output function 358
9.3.5 Cascade connection 362
9.4 Control Registers 363
9.4.1 Timer control registers (TMC0, TMC1) 363
9.4.2 Timer output control registers (TOC0, TOC1) 365
9.5 Interrupt Requests 366
9.5.1 Interrupt requests from timer unit 366
9.5.2 Capture trigger dual function external interrupt 366
9.6 Caution 367

CHAPTER 10 REAL-TIME OUTPUT PORT FUNCTION 369
10.1 Features 369
10.2 Configuration 369
10.3 Operation 371
10.3.1 Basic operation 371
10.3.2 Delay function 373
10.3.3 Use with macro service 374
10.3.4 Application example 375
10.3.5 Port access operations 376
10.4 Control Registers 377
10.4.1 Port 7 mode control register (PMC7) 377
10.4.2 Real-time output port control register (RTPC) 378
10.4.3 Real-time output port delay specification register (RTPD) 380
10.4.4 Delay counter 380

CHAPTER 11 PWM OUTPUT FUNCTION 381
11.1 Features 381
11.2 Configuration 381
11.3 Operation 383
11.3.1 Setting 383
11.3.2 Basic operation 383
11.3.3 PWM output operations 384
11.4 Control Registers 386
11.4.1 PWM control register (PWMC) 386
11.4.2 PWM register (PWM) 386

CHAPTER 12 WATCHDOG TIMER FUNCTION 387
12.1 Features 387
12.2 Configuration and Operation 387
12.3 Control Registers 389
12.3.1 Watchdog timer mode register (WDM) 389

CHAPTER 13 A/D CONVERTER FUNCTION 391
13.1 Features 391
13.2 Configuration 392

13.3 Operation 394
13.3.1 Basic operations of A/D converter 394
13.3.2 Select mode 396
13.3.3 Scan mode 397
13.4 Control Registers 399
13.4.1 A/D converter mode register (ADM) 399
13.4.2 A/D conversion result register (ADCR0-ADCR3) 400
13.5 Interrupt Request 400
13.6 Caution 400
13.6.1 Executing the STOP instruction 400
13.6.2 A/D conversion sequence initialization 400
13.6.3 Loss of conversion result 400

CHAPTER 14 PORT FUNCTIONS 401
14.1 Features 401
14.2 Configuration 401
14.2.1 When specified output port (PMXn = 0) 403
14.2.2 When specified input port (PMXn = 1) 405
14.2.3 Control specification (PMXn = 1) 406
14.3 Port Pins Functions 407
14.3.1 Port 0 (P00 to P07) 407
14.3.2 Port 1 (P10 to P16) 408
14.3.3 Port 2 (P20 to P25) 409
14.3.4 Port 3 (P30 to P36) 410
14.3.5 Port 4 (P40 to P47) 411
14.3.6 Port 5 (P50 to P52) 413
14.3.7 Port 6 (P60 to P63) 414
14.3.8 Port 7 (P70 to P77) 415
14.3.9 Port 8 (P80, P81) 417

CHAPTER 15 STANDBY FUNCTION 419
15.1 Features 419
15.2 Standby Modes 419
15.3 Control Registers 420
15.3.1 Standby control register (STBC) 420
15.4 HALT Mode 420
15.4.1 HALT mode setting and operation 420
15.4.2 HALT mode release 420
15.5 STOP Mode 422
15.5.1 STOP mode setting and operation 422
15.5.2 STOP mode release 422

CHAPTER 16 CLOCK GENERATOR 423
16.1 Features 423
16.2 Configuration 423
16.3 Operation 424
16.3.1 Clock oscillation circuit 424
16.3.2 Time base counter (TBC) 425

16.4 Processor Control Register (PRC) 426
16.5 Clock Switching Timing 428

CHAPTER 17 SOFTWARE INTERVAL TIMER FUNCTION 429
17.1 Configuration 429
17.2 Control Registers 429
17.3 Interrupt Request 430

CHAPTER 18 CODEC FUNCTIONS 431
18.1 Features 431
18.2 Encoding/Decoding Methods of Binary Image Data 434
18.2.1 Run-length encoding 434
18.2.2 MH encoding 437
18.2.3 MR encoding 441
18.2.4 MMR encoding 448
18.3 Memory Map 449
18.4 Encoding/Decoding Conversion Tables 451
18.4.1 Encoding conversion table 451
18.4.2 Decoding conversion table 453
18.4.3 Decoding conversion table (for MH decoding instruction) 455
18.4.4 Encoding conversion table (for MH encoding instruction) 465
18.5 Processing Flow 472
18.6 Descriptions of Functions of Each Instruction 474
18.6.1 Data transmit instruction 474
18.6.2 Turning point table create instruction 477
18.6.3 MH encode instruction 479
18.6.4 MR encode instruction 481
18.6.5 EOL detect instruction 483
18.6.6 1-bit detect instruction 486
18.6.7 MH decode turning point table create instruction 488
18.6.8 MR decode turning point table create instruction 491
18.6.9 Pixel data create instruction 494
18.7 Processing Time 496

CHAPTER 19 RESET FUNCTION 499

APPENDIX A LIST OF MNEMONIC 503

APPENDIX B OPERATING PRECAUTIONS 537
B.1 Interrupt Function 537
B.2 DMA Function 538
B.3 Serial Interface Function (UART) 541
B.4 Serial Interface Function (CSI) 542
B.5 Timer Function 542
B.6 Real-Time Output Port Function 543
B.7 A/D Converter Function 543
B.8 Codec Function 543

APPENDIX C REGISTER INDEX 545

LIST OF FIGURES (1/5)

Figure No.	Title	Page
1-1.	Positioning of V55PI	1
2-1.	Input/Output Circuits Classified by Types	34
3-1.	Register Bank Configuration	39
3-2.	PSW Bit Configuration	45
3-3.	Basic Memory Space	51
3-4.	Expansion Memory Space	52
3-5.	Register File Space	65
3-6.	I/O Map (64 Kbytes)	66
3-7.	Concept of Segment Method	67
3-8.	Relationship among the Segment Register, Offset Address, and Physical Address	68
3-9.	Relationship among the Expansion Segment Register, Offset Address, and Physical Address	68
3-10.	Relationship between Segment Register, Segment, and Memory Space	70
3-11.	Dynamic Relocation	73
3-12.	Memory Address Operation	74
4-1.	Memory Divided Control	92
4-2.	Memory Block Status after Reset	93
4-3.	Memory Read Cycle Wait Insertion by READY Pin (When there is no address wait and data wait by PWC register is 1)	96
4-4.	Memory Write Cycle Wait Insertion by READY Pin (When there is no address wait and data wait by PWC register is 1)	97
4-5.	Example of DRAM Connection Circuit (Address Wait Insertion)	100
4-6.	Refresh Cycle Timing Chart (TAW is valid only when address wait is inserted)	104
4-7.	Example of Pseudo-SRAM Connection Circuit (Address Wait Insertion)	105
4-8.	Memory Read Cycle	108
4-9.	Memory Read Cycle with Data Wait (1 Wait) Inserted	109
4-10.	Memory Read Cycle with Address Wait Inserted	110
4-11.	Memory Read Cycle with Data Wait (1 Wait) and Address Wait Inserted	111
4-12.	Memory Write Cycle	112
4-13.	Memory Write Cycle with Data Wait (1 Wait) Inserted	113
4-14.	Memory Write Cycle with Address Wait Inserted	114
4-15.	Memory Write Cycle with Data Wait (1 Wait) and Address Wait Inserted	115
4-16.	I/O Read Cycle	116
4-17.	I/O Read Cycle with Data Wait (1 Wait) Inserted	117
4-18.	I/O Write Cycle	118
4-19.	I/O Write Cycle with Data Wait (1 Wait) Inserted	119
5-1.	Maskable Interrupt Service Modes	127
5-2.	Interrupt Acknowledge Operation (Performed in Sequence ① → ④)	132
5-3.	Vector Table Configurations	132

LIST OF FIGURES (2/5)

Figure No.	Title	Page
5-4.	Vector Table and Register Bank Number	133
5-5.	Register Bank Switching Sequence (Register Bank Switching by Interrupt)	135
5-6.	Register Bank Restoration Sequence	136
5-7.	Sequence for Register Bank Switching by TSKSW Instruction	138
5-8.	Interrupt Servicing Efficiency with Macro Service	139
5-9.	Basic Configuration of Macro Service Control Word	142
6-1.	$\overline{\text{DMAAK}}$ Output Example (Single Transfer Mode) (Memory → I/O Transfer)	201
6-2.	DMA Transfer Timing Example (Single Transfer Mode) (Memory → I/O Transfer)	202
6-3.	Channel Configuration	203
6-4.	Single Transfer Mode Transfer Timing	207
6-5.	Demand Release Mode Transfer Timing	207
6-6.	Single Step Mode Transfer Timing	208
6-7.	Burst Mode Transfer Timing	208
6-8.	Intelligent DMA Mode-1 (Ring Buffer Method)	210
6-9.	Buffer Memory in Decrement Mode (Intelligent DMA Mode-1)	212
6-10.	Intelligent DMA Mode-2 (Counter Control Method)	215
6-11.	Buffer Memory in Decrement Mode (Intelligent DMA Mode-2)	217
6-12.	Next Address Specification Mode	219
6-13.	2-Channel Operation Mode (Stop Mode)	222
6-14.	2-Channel Operation Mode (Repeat Mode)	224
6-15.	Memory-to-Memory Transfer Mode (Stop Mode)	226
6-16.	Memory-to-Memory Transfer Mode (Repeat Mode)	228
6-17.	Memory → External I/O	248
6-18.	External I/O → Memory	250
6-19.	Memory → Internal I/O (SFR)	252
6-20.	Internal I/O (SFR) → Memory	253
6-21.	Memory ↔ Memory	254
6-22.	Memory → SFR (Memory Access)	256
6-23.	SFR (Memory Access) → Memory	258
7-1.	Baud Rate Generator Block Diagram	269
7-2.	UART Block Diagram	274
7-3.	Basic Data Format	276
7-4.	Example of Connection of Multiple Systems	278
7-5.	Data Format When Wake-Up Function is Selected	278
7-6.	UART Reception Error Interrupts Generation Timing	287
7-7.	UART Reception Interrupts Generation Timing	288
7-8.	UART Transmission Interrupt Generation Timing	289
7-9.	Clocked Serial Interface Block Diagram	293
7-10.	Example of IOE Mode System Configuration	296
7-11.	IOE Mode Transfer Timing	297
7-12.	Example of SBI Mode System Configuration	299

LIST OF FIGURES (3/5)

Figure No.	Title	Page
7-13.	Address Transmission from Master Device to Slave Device	300
7-14.	Command Transmission from Master Device to Slave Device	300
7-15.	Data Transmission from Master Device to Slave Device	301
7-16.	Data Transmission from Slave Device to Master Device	301
7-17.	$\overline{SCK0}$ Pin and Bus Line	310
7-18.	$\overline{SCK0}$ Pin and Bus Line (Command Condition Detection)	310
7-19.	RELT, CMDT, RELD, and CMDD Operations	312
7-20.	ACKT Operation	312
7-21.	ACKE Operation	313
7-22.	ACKD Operation	314
7-23.	BSYE Operation	315
8-1.	Parallel Interface Block Diagram	318
8-2.	Timing Chart (Output Mode)	322
8-3.	Timing Chart (STBT=0, BSYT=0)	335
8-4.	Timing Chart (STBT=1, BSYT=0)	335
8-5.	Timing Chart (STBT=1, BSYT=1)	335
8-6.	Timing Chart (STBT=0, BSYT=1)	336
8-7.	Timing Chart (ACKT1, ACKT0=00)	337
8-8.	Timing Chart (ACKT1, ACKT0=01)	337
8-9.	Timing Chart (ACKT1, ACKT0=10)	337
8-10.	Timing Chart (ACKT1, ACKT0=00)	338
8-11.	Timing Chart (ACKT1, ACKT0=01)	338
8-12.	Timing Chart (ACKT1, ACKT0=10)	338
8-13.	Timing Chart (ACKT1, ACKT0=00)	339
8-14.	Timing Chart (ACKT1, ACKT0=01)	339
8-15.	Timing Chart (ACKT1, ACKT0=10)	339
8-16.	Timing Chart (ACKT1, ACKT0=00)	340
8-17.	Timing Chart (ACKT1, ACKT0=01)	340
8-18.	Timing Chart (ACKT1, ACKT0=10)	340
8-19.	Timing Chart (ACKT1, ACKT0=00)	341
8-20.	Timing Chart (ACKT1, ACKT0=01)	341
8-21.	Timing Chart (ACKT1, ACKT0=10)	341
8-22.	Timing Chart (ACKT1, ACKT0=00)	342
8-23.	Timing Chart (ACKT1, ACKT0=01)	342
8-24.	Timing Chart (ACKT1, ACKT0=10)	342
8-25.	Timing Chart (ACKT1, ACKT0=00)	343
8-26.	Timing Chart (ACKT1, ACKT0=01)	343
8-27.	Timing Chart (ACKT1, ACKT0=10)	343
8-28.	Timing Chart (ACKT1, ACKT0=00, ACKM=0)	344
8-29.	Timing Chart (ACKT1, ACKT0=01, ACKM=0)	344
8-30.	Timing Chart (ACKT1, ACKT0=10, ACKM=0)	344
8-31.	Timing Chart (ACKT1, ACKT0=00, ACKM=0)	345

LIST OF FIGURES (4/5)

Figure No.	Title	Page
8-32.	Timing Chart (ACKT1, ACKT0=01, ACKM=0)	345
8-33.	Timing Chart (ACKT1, ACKT0=10, ACKM=0)	345
9-1.	Timer Unit Block Diagram	349
9-2.	Free Running Timer Operation (Timer 0)	351
9-3.	Interval Timer Operation (When CM10 is used as the compare register)	352
9-4.	Capture Operation (Timer 0)	354
9-5.	Compare Operation (Timer 0)	356
9-6.	When Timer 0 is Used as Interval Timer	357
9-7.	Toggle Output by Timer 0	360
9-8.	Set/Reset Output by Timer 2	361
9-9.	TO30 Output Waveform by Cascade Connection	362
10-1.	Real-Time Output Port Block Diagram	370
10-2.	Real-Time Output Port Operation Timing	372
10-3.	Real-Time Output Port Output Operation (Delay Mode)	374
10-4.	Stepping Motor Control by Real-Time Output Port	375
11-1.	PWM Unit Block Diagram	382
11-2.	PWM Output Operation Timing	383
11-3.	PWM Output Operation Timing (CE = 0 → 1)	384
11-4.	PWM Output Operation Timing (CE = 1 → 0)	385
11-5.	PWM Output Operation Timing (When the ALV bit is changed)	385
12-1.	$\overline{\text{RESET}}$ Pin Connection Example	387
12-2.	Watchdog Timer Block Diagram	389
13-1.	A/D Converter Block Diagram	393
13-2.	A/D Converter Basic Operation	395
13-3.	Select Mode Operation Timing	397
13-4.	Scan Mode Operation Timing	398
14-1.	Ports 0, 2 to 5, 7, 8 Configuration	402
14-2.	Output Port Specification Port	403
14-3.	Input Port Specification Port	405
14-4.	Control Specification Port	406
14-5.	Port 1 Block Diagram	408
14-6.	Port 6 Block Diagram	415
15-1.	STOP Mode Release by NMI Pin Input	422
16-1.	Clock Generator	423
16-2.	Clock Generator External Circuits	424

LIST OF FIGURES (5/5)

Figure No.	Title	Page
16-3.	Time Base Counter (TBC) Configuration	425
16-4	System Clock Switching Timing	428
17-1.	Software Interval Timer Configuration	429
18-1.	MH, MR Encoding Process Flow	432
18-2.	MH, MR Decoding Process Flow	433
18-3.	Sample Picture	435
18-4.	Bit Map Data	435
18-5.	Run-length Encoding	436
18-6.	Encoding Example of One Line	439
18-7.	MH Encoding of One Page	440
18-8.	Correlation Between Vertical Directions of Image	441
18-9.	Reference Points for Encoding	442
18-10.	Path Mode Example	442
18-11.	Vertical Mode Example (When a1 lies at the 2nd pixel to the right of b1)	443
18-12.	Horizontal Mode Example	444
18-13.	Processing Example of Starting Point of Line	445
18-14.	Processing Example of Ending Point of Line	445
18-15.	K Parameter (When K = 4)	446
18-16.	MR Encoding of One Page	447
18-17.	MMR Encoding of One Page	448
18-18.	Encode Instructions and Data in Each Memory	450
18-19.	Decode Instructions and Data in Each Memory	450
18-20.	One-Line Data Encoding Flow	472
18-21.	One-Line Data Decoding Flow	473
18-22.	Processing Time of MH Encoding/Decoding Instruction	496
18-23.	Processing Time of MR Encoding/Decoding Instruction	497

LIST OF TABLES (1/3)

Table No.	Title	Page
1-1.	Comparison of V55PI and V25 Functions	3
2-1.	Port Pin and Control Pin Statuses	16
2-2.	Non-Port Pin Statuses	18
2-3.	Port 0 Operation (n = 0 to 7)	19
2-4.	Port 1 Operation	19
2-5.	Port 2 Operation (n = 0 to 5)	20
2-6.	Port 3 Operation (n = 0 to 6)	21
2-7.	Port 4 Operation (n = 0 to 7)	23
2-8.	Port 5 Operation (n = 0 to 2)	24
2-9.	Port 6 Operation	26
2-10.	Port 7 Operation (n = 0 to 7)	26
2-11.	Port 8 Operation (n = 0, 1)	27
2-12.	$\overline{WRL}$ and $\overline{WRH}$ Pins Operation in the Memory Write Cycle (Using 16-bit External Data Bus Width)	30
3-1.	General Register Offset	40
3-2.	Pointer and Index Register Offset	41
3-3.	Segment Register Offset	42
3-4.	Expansion Segment Register Offset	44
3-5.	List of Special-Function Registers	54
3-6.	Relationship between Segment Register and Offset Address Specification	69
4-1.	Wait in SFR Access (Above 1 Wait)	98
5-1.	Interrupt Sources	123
5-2.	Interrupt Requests and Macro Service Control Word Addresses	143
5-3.	Macro Service Operating Modes	145
5-4.	EVENT Mode Operation Specification	149
5-5.	Data Byte/Word Specification	157
5-6.	BLKTRS Mode Operation Specification	160
5-7.	Memory Space Address Mode Specification	160
5-8.	BLKTRS-C Mode Operation Specification	163
5-9.	Memory Space Address Mode Specification	163
5-10.	Timer Compare Register Set Data Table Address Update Setting	165
5-11.	Timer Compare Register Set Data Manipulation Specification	165
5-12.	Interrupt Request Signals and SFRs	168
5-13.	Real-Time Output Port Write Specification	168
5-14.	Software Interrupts	180
5-15.	Macro Service Processing Time (Unit: Clock)	195
6-1.	DMA Transfer Target and Transfer Method	200
6-2.	Possible Combination of Transfer Target and Start Source	205

LIST OF TABLES (2/3)

Table No.	Title	Page
6-3.	Transfer Modes	208
6-4.	List of Operating Modes	209
6-5.	Relationship Between IOPn and DPTCn	214
6-6.	Registers in Each Operating Mode	242
6-7.	External Data Bus Width and DMA Transfer Objects	261
7-1.	Supported Protocols	265
7-2.	Control Registers	268
7-3.	Control Pins	268
7-4.	Values Set in BRG Register	271
7-5.	Receive Error Causes	287
7-6.	UART Interrupt Requests Connectable to DMA Requests	290
7-7.	Clocked Serial Interface Pins	292
7-8.	Function Specification by Control Register (SO/SB0 Pins)	294
7-9.	Function Specification by Control Register (SI/SB1 Pins)	294
7-10.	Function Specification by Control Register (SO Pin)	294
7-11.	Function Specification by Control Register (SI Pin)	295
7-12.	Function Specification by Control Register ($\overline{SCK}$ Pin)	295
7-13.	Signals in SBI Mode	303
8-1.	Specification of Trigger Pins for Interrupt Request and DMA Request Generation (Output Mode)	321
8-2.	Specification of $\overline{ACK}$ Signal Output (When MB bit is reset)	325
8-3.	Setting of ACKM, ACKT1, ACKT0 and Timing of BUSY Signal and $\overline{ACK}$ Signal	325
8-4.	Typical PAI Set Values and $\overline{ACK}$ Output Width	332
9-1.	Types of Timer Units	347
9-2.	Functions of Each Timer	350
9-3.	Timer Count Time	351
10-1.	RTPD Set Value and Delay Time	373
10-2.	Port 7 Access Objective	376
10-3.	Data Transfer Timing to Real-Time Output Port (RTP)	379
11-1.	PWM Register Value and Duty Ratio	386
13-1.	A/D Conversion Time	396
14-1.	Port 0 Operation (n = 0 to 7)	407
14-2.	Port 1 Operation	408
14-3.	Port 2 Operation (n = 0 to 5)	409
14-4.	Port 3 Operation (n = 0 to 6)	410
14-5.	Port 4 Operation (n = 0 to 7)	412

LIST OF TABLES (3/3)

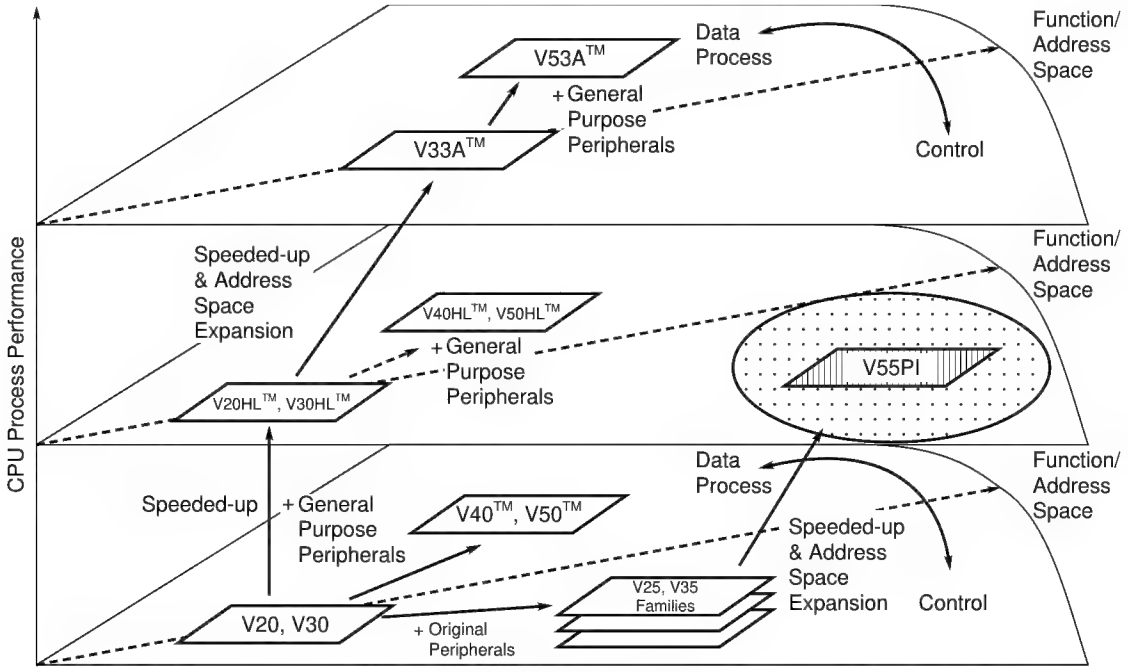
Table No.	Title	Page
14-6.	Port 5 Operation (n = 0 to 2)	413
14-7.	Port 6 Operation	414
14-8.	Port 7 Operation (n = 0 to 7)	415
14-9.	Port 8 Operation (n = 0, 1)	417
15-1.	HALT/STOP Mode Operation States	419
15-2.	HALT Mode Release by Macro Service Request	421
18-1.	Run-length Code	438
18-2.	Vertical Mode Code	443
18-3.	Codec Instructions Addable with Expansion Segment Override Prefix	449
18-4.	MHDEC Instruction End Status	490
18-5.	MRDEC Instruction End Status	493
19-1.	Hardware States After Reset	499
A-1.	List of Mnemonics	504

CHAPTER 1 GENERAL DESCRIPTION

The V55PI is a microprocessor which integrates a 16-bit CPU, a RAM, a serial interface, a parallel interface, an A/D converter, a timer, a DMA controller, an interrupt controller, and other relevant units in a single chip.

The V55PI is software upward compatible with single-chip microcontrollers V25, V35 and microprocessor V20, V30. To provide a migration path from the V25, the V55PI has more advanced functions and higher performance, and is best suited for data processing systems including mechanical controls, such as printers and facsimile.

★ **Figure 1-1. Positioning of V55PI**



16-Bit V Series Products Line-up

1.1 Features

The V55PI has the following functions.

- 16-bit internal architecture:
  Selectable external data bus width (16-bit or 8-bit)
- Software-compatible with V20, V30 (native mode) and V25, V35 (additional instructions available)
- Minimum instruction cycle: 160 ns at 12.5 MHz (externally at 25 MHz)
  125 ns at 16 MHz (externally at 32 MHz)
- Address space: 16 Mbytes
  - Basic memory space: 1 Mbyte
  - Expansion memory space: 16 Mbytes
- Register file space (internal RAM): 512 bytes/16 register banks
- I/O space: 64 Kbytes
- Automatic wait control by dividing memory space into variable size (6 blocks max.)
- I/O line (input-only port: 11 bits, I/O port: 42 bits)
- DMA controller (DMAC): Configuration possible with a maximum of 4 channels
  - 4 types of DMA transfer modes (single transfer, demand release, single step, burst)
  - Intelligent DMA modes 1 and 2
- Serial interface: 2 channels
  - Selection of either asynchronous mode (UART) or clocked mode (CSI)
- Parallel interface: 8 bits
  - Centronics data input/output and general data input/output
- A/D converter (8-bit resolution): 4 channels
- Real-time output port: 4 bits × 2 channels, or 8 bits × 1 channel
- PWM (Pulse Width Modulation) signal output function (8 bits)
- Interrupt controller
  - Multiple interrupt service control by programmable priority (4 levels)
  - Three types of interrupt response methods
    Vectored interrupt, register bank switching, and macro service
- 16-bit timer counter: 4 channels
- Watchdog timer function
- Software interval timer (16 bits)
- Address block wait insertion and RAS/CAS switching timing generation
- DRAM and pseudo-SRAM refresh function
- Standby function (STOP mode and HALT mode)
- On-chip clock generator

1.2 Comparison of V55PI and V25 Functions

Table 1-1 compares the V55PI and V25 functions.

Table 1-1. Comparison of V55PI and V25 Functions

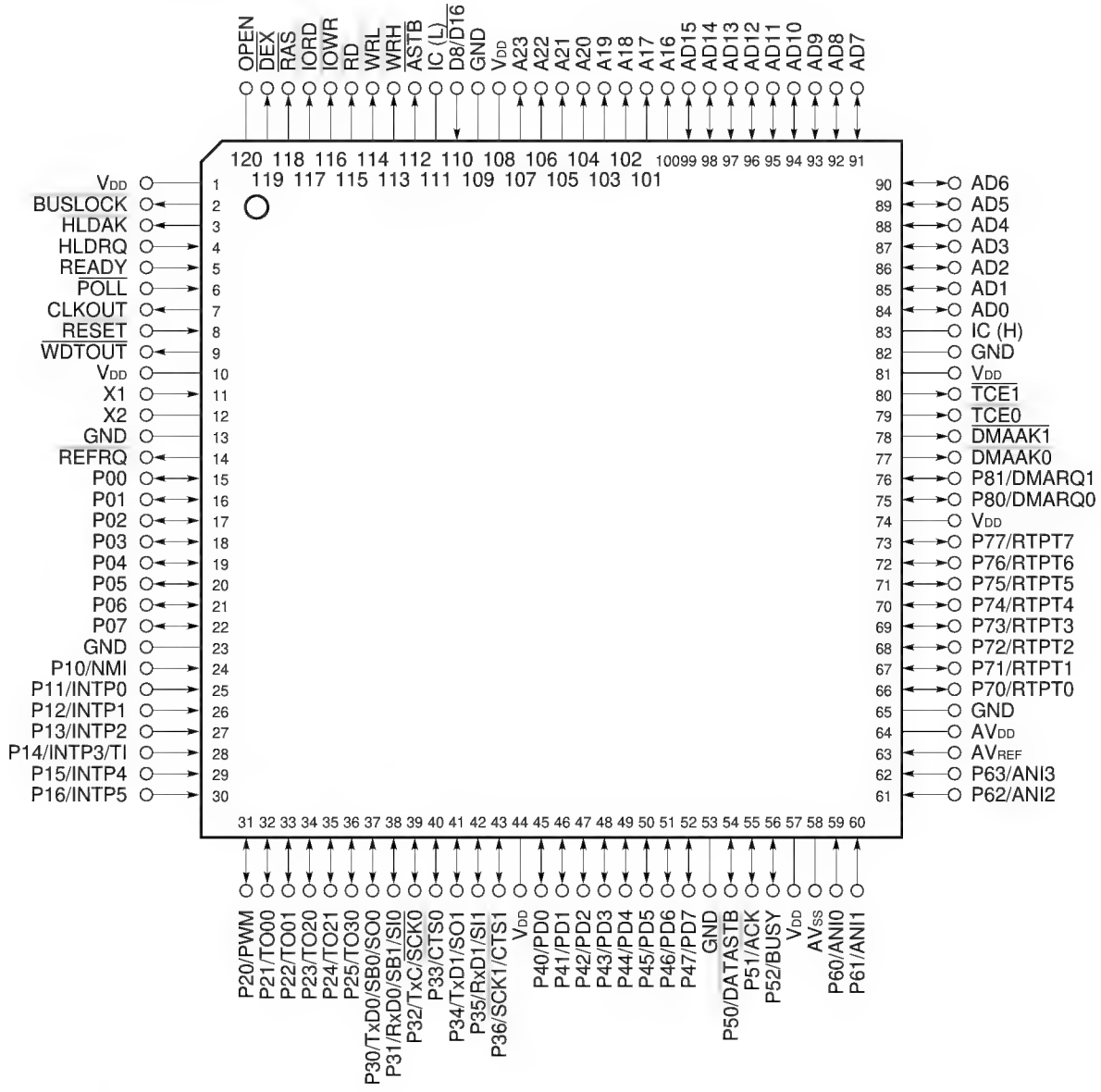
Item			V25	**V55PI**
CPU	Address space	Memory	1 Mbyte	16 Mbytes
		I/O	64 Kbytes	64 Kbytes
		Register file	None	512 bytes
	Register bank (with internal RAM)		8 banks/256 bytes	16 banks/512 bytes
	Bus configuration (internal/external: bit)		16/8	Selection of 16/16 or 16/8
	Software successability		V30 upward compatible	V25 and V30 upward compatible
Peripheral Hardware	Serial interface	No. of channels	2 channels	2 channels
		Interface mode	Asynchronous mode (UART) I/O interface mode	UART Clocked mode (SBI/IOE mode)
	Parallel interface	No. of channels	None	1 channel (8 bits)
		Specifications		Centronics applicable
	DMA function	No. of channels	2 channels	4 channels
		Transfer mode	4 modes	4 modes
		Operating mode		5 modes
	Interrupt function	Level	8 levels	4 levels
		Response method	Vectored interrupt, Register bank switching, Macro service (2 modes)	Vectored interrupt, Register bank switching, Macro service (7 modes)
	General-purpose port		4 (input-only) 20 (input/output)	11 (input-only) 42 (input/output)
	Real-time output port		None	2 channels (4 bits)
	Timer function		2 channels (16 bits)	5 channels (16 bits) (software interval timer included)
	PWM output function		None	1 channel (8 bits)
	A/D converter		None	4 channels (8 bits)
	Programmable wait function		Memory division into 7 portions (with fixed address)	Memory division into 6 portions (with variable address)
	Refresh function (Refresh address)		Available (9 bits)	Available (20 bits)
	Watchdog timer function		None	Available
	Standby function		STOP, HALT	STOP, HALT

★ 1.3 Ordering Information

Part Number	Package	Maximum Operating Frequency (MHz)
μPD70433GD-12-5BB	120-pin plastic QFP (28 × 28 mm)	12.5
μPD70433GD-16-5BB	120-pin plastic QFP (28 × 28 mm)	16
μPD70433R-12	132-pin ceramic PGA	12.5
μPD70433R-16	132-pin ceramic PGA	16
μPD70433GJ-12-3EB	120-pin plastic QFP (fine-pitch) (20 × 20 mm)	12.5
μPD70433GJ-16-3EB	120-pin plastic QFP (fine-pitch) (20 × 20 mm)	16

1.4 Pin Configuration (Top View)

(1) 120-pin plastic QFP (28 × 28 mm),
120-pin plastic QFP (fine-pitch) (20 × 20 mm)
μPD70433GD–xx–5BB
μPD70433GJ–xx–3EB

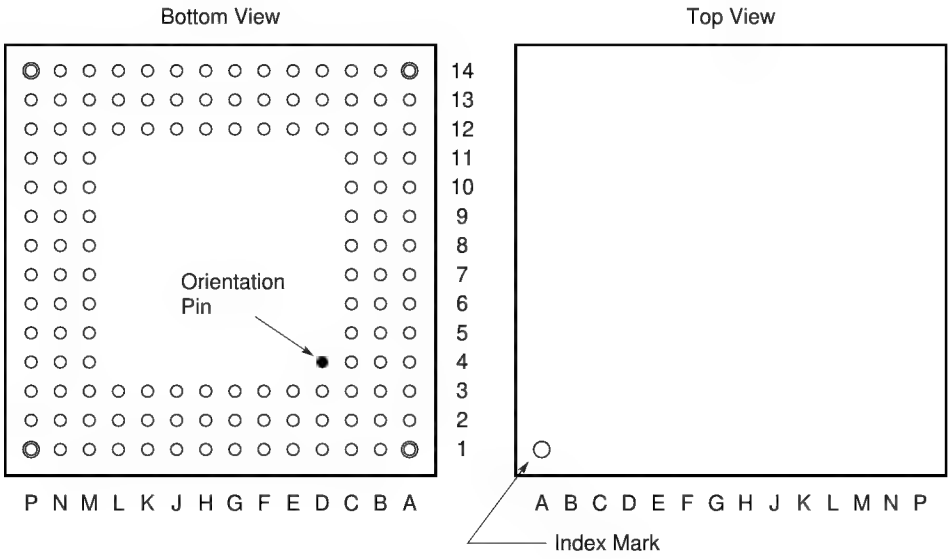


Remark IC: Internally Connected

Cautions 1. Connect each IC (H) pin externally to V_{DD} via a resistor (1 kΩ to 10 kΩ).
2. Connect each IC (L) pin externally to GND via a resistor (1 kΩ to 10 kΩ).
3. Do not connect anything to the OPEN pins.

(2) 132-pin ceramic PGA

μPD70433R–××



Remark Orientation pins are not counted as pins.

No.	Signal Name	Port	No.	Signal Name	Port	No.	Signal Name	Port
A1	ANI1	P61	B5	PD7	P47	C9	$\overline{CTS0}$	P33
A2	AV_{SS}	—	B6	PD5	P45	C10	TO30	P25
A3	$\overline{ACK}$	P51	B7	PD2	P42	C11	TO00	P21
A4	$\overline{DATASTB}$	P50	B8	PD0	P40	C12	NC	—
A5	PD6	P46	B9	RxD1/SI1	P35	C13	INTP4	P15
A6	PD4	P44	B10	RxD0/SB1/SI0	P31	C14	INTP0	P11
A7	PD1	P41	B11	TO21	P24	D1	RTPT2	P72
A8	NC	—	B12	TO01	P22	D2	GND	—
A9	$\overline{SCK1}/\overline{CTS1}$	P36	B13	NC	—	D3	ANI3	P63
A10	TxD1/SO1	P34	B14	INTP3/TI	P14	D12	INTP5	P16
A11	TxC/$\overline{SCK0}$	P32	C1	RTPT1	P71	D13	INTP2	P13
A12	TxD0/SB0/SO0	P30	C2	AV_{REF}	—	D14	NMI	P10
A13	TO20	P23	C3	NC	—	E1	RTPT5	P75
A14	PWM	P20	C4	NC	—	E2	RTPT3	P73
B1	AV_{DD}	—	C5	V_{DD}	—	E3	RTPT0	P70
B2	ANI2	P62	C6	GND	—	E12	INTP1	P12
B3	ANI0	P60	C7	PD3	P43	E13	GND	—
B4	BUSY	P52	C8	V_{DD}	—	E14	—	P06

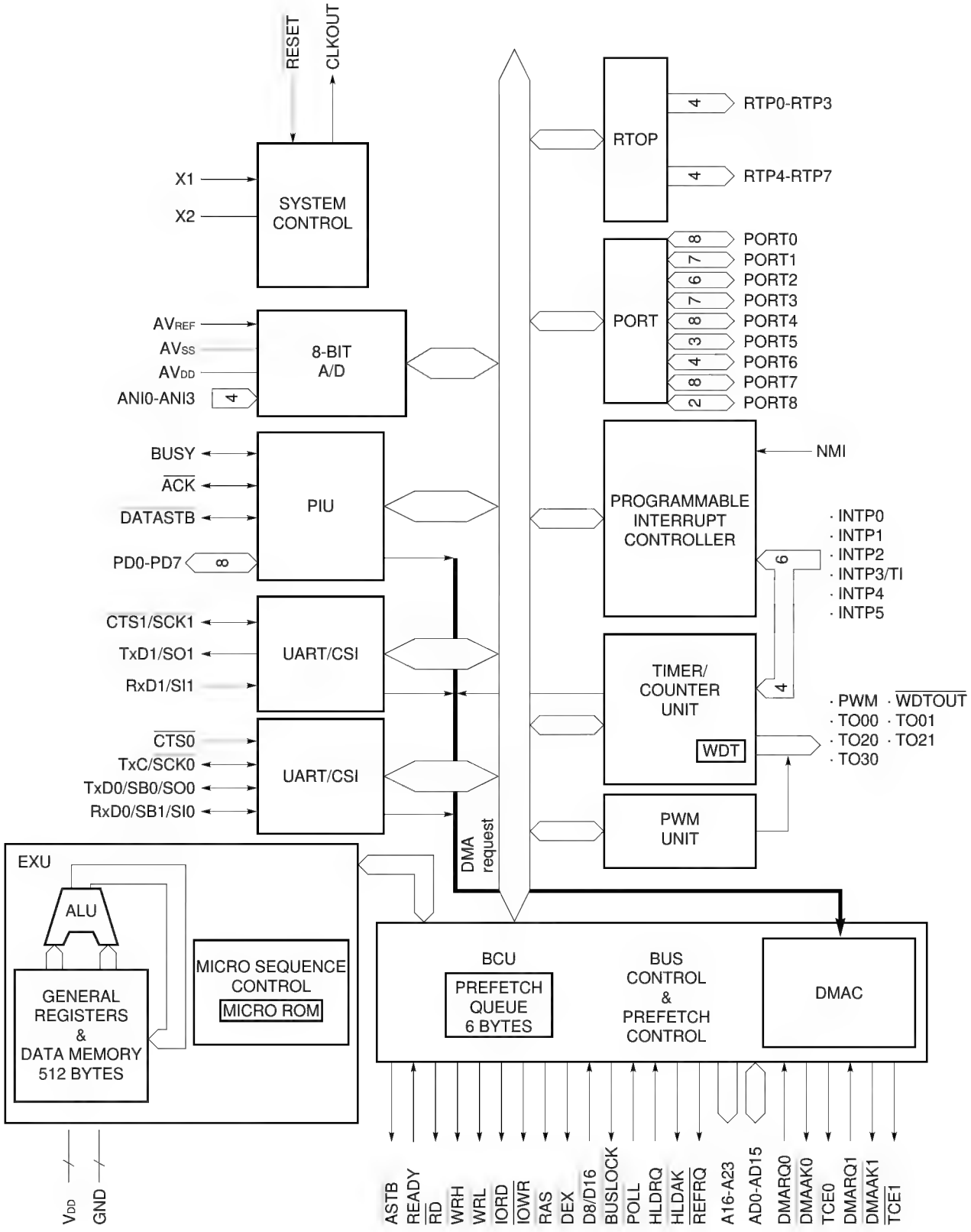
No.	Signal Name	Port	No.	Signal Name	Port	No.	Signal Name	Port
F1	RTPT7	P77	K3	AD2	—	N3	AD9	—
F2	RTPT6	P76	K12	$\overline{\text{POLL}}$	_	N4	AD11	—
F3	RTPT4	P74	K13	$\overline{\text{WDTOUT}}$	—	N5	AD14	—
F12	—	P07	K14	X1	—	N6	A18	—
F13	—	P05	L1	AD0	—	N7	A21	—
F14	—	P04	L2	AD3	—	N8	A23	—
G1	NC	—	L3	AD6	—	N9	D8/$\overline{\text{D16}}$	—
G2	DMARQ0	P80	L12	$\overline{\text{BUSLOCK}}$	—	N10	$\overline{\text{ASTB}}$	—
G3	V_{DD}	—	L13	READY	—	N11	$\overline{\text{IOWR}}$	—
G12	—	P03	L14	$\overline{\text{RESET}}$	—	N12	$\overline{\text{DEX}}$	—
G13	—	P02	M1	AD1	—	N13	V_{DD}	—
G14	—	P01	M2	AD5	—	N14	HLDRQ	—
H1	DMARQ1	P81	M3	NC	—	P1	AD7	—
H2	$\overline{\text{DMAAK0}}$	—	M4	AD8	—	P2	AD10	—
H3	$\overline{\text{DMAAK1}}$	—	M5	AD12	—	P3	AD13	—
H12	$\overline{\text{REFRQ}}$	—	M6	A16	—	P4	AD15	—
H13	—	P00	M7	A20	—	P5	A17	—
H14	NC	—	M8	V_{DD}	—	P6	A19	—
J1	$\overline{\text{TCE0}}$	—	M9	$\overline{\text{WRH}}$	—	P7	NC	—
J2	$\overline{\text{TCE1}}$	—	M10	$\overline{\text{IORD}}$	—	P8	A22	—
J3	GND	—	M11	NC	—	P9	GND	—
J12	V_{DD}	—	M12	NC	—	P10	IC (L)	—
J13	X2	—	M13	$\overline{\text{HLDAK}}$	—	P11	$\overline{\text{WRL}}$	—
J14	GND	—	M14	CLKOUT	—	P12	$\overline{\text{RD}}$	—
K1	V_{DD}	—	N1	AD4	—	P13	$\overline{\text{RAS}}$	—
K2	IC (H)	—	N2	NC	—	P14	OPEN	—

Remark IC: Internally Connected
NC: Non-Connection

Cautions 1. Connect each IC (H) pin externally to V_{DD} via a resistor (1 kΩ to 10 kΩ).
2. Connect each IC (L) pin externally to GND via a resistor (1 kΩ to 10 kΩ).
3. Do not connect anything to the OPEN pins.

1.5 Internal Block

1.5.1 Internal block diagram



1.5.2 Internal block configuration

- **Bus Control Unit (BCU)**
  The BCU controls the bus. In the BCU the necessary internal/external bus cycles are started in accordance with the physical address obtained with the execution unit (EXU).

- **Execution Unit (EXU)**
  The EXU controls address calculations, arithmetic logic calculations, and data transfer with microprograms (firmware to control the micro sequencer based on the operation code decoding results.)
  The EXU incorporates a 512-byte RAM (corresponding to the register file space.)

- **Interrupt Controller (INTC)**
  Interrupt requests from the on-chip peripheral hardware and external hardware are serviced either by the vectored interrupt, switching the register bank, or with macro service. Further, programmable 4-level interrupt priority can be controlled and multiple processing control can be executed on interrupt sources.

- **DMA Controller (DMAC)**
  The DMAC is a general DMA controller and can handle a 16-Mbyte memory space linearly. The operating modes consist of a memory-to-memory transfer mode, an intelligent DMA (ring buffer method and counter control method) mode, a next address specify mode, and 2-channel operating mode.

- **UART/Clocked Serial Interface (UART/CSI)**
  The asynchronous serial interface (UART) for data synchronization with the start/stop bit and the clocked serial interface (CSI) are supported and either interface can be selected for use.
  The clocked serial interface can employ either the serial bus interface mode (SBI) or the 3-wire serial I/O mode.

- **Parallel Interface Unit (PIU)**
  Performs input/output in 8-bit units using strobe signal synchronization and supports the Centronics interface and general-purpose parallel data communication functions.

- **A/D Converter Unit (8-bit A/D)**
  This is a successive conversion 8-bit A/D converter having four analog inputs. It is equipped with four A/D conversion result registers.

- **Timer/Counter Unit (TCU)**
  This unit incorporates a 16-bit timer/counter and can be used as the interval timer, the free-running timer, or the event counter.

- **Pulse Width Modulation Unit (PWM)**
  This unit is provided with an 8-bit accuracy PWM (pulse width modulation) signal output function.

- **Watchdog Timer (WDT)**
  An 8-bit watchdog timer is incorporated to detect an inadvertent program overrun and system errors. The WDT is equipped with the $\overline{\text{WDTOUT}}$ pin to externally notify the generation of a watchdog timer interrupt.

- **Port (PORT)**
  53 port pins are installed. Port pin and control pin functions can be selected for use.

- **Real-time Output Port (RTOP)**
  This is a real-time output port regarding the interrupt from timer 0 as a trigger. It can output 8-bit buffer register contents in 4-bit or 8-bit units at a programmable interval.

- **Clock Generator (CG)**
  The CG generates 1/2, 1/4, 1/8, and 1/16 specific frequencies of the crystal resonator/ceramic resonator connected to the X1 and X2 pins, and supplies them as CPU operation clocks.

- **Software Interval Timer (SIT)**
  A 16-bit software interval timer is incorporated as one for software timer or watch functions. It can set interval interrupts by selecting the input clock (count clock) and setting the software timer counter compare register.

CHAPTER 2 PIN FUNCTIONS

The V55PI pins can be functionally divided into port pins and non-port pins.

2.1 List of Pin Functions

2.1.1 Port functions

Pin Name	Input/Output	Function	Alternate Functions
P00 to P07	Input/output	Port 0 Input/output specifiable in bit units 8-bit input/output port	—
P10[Note]	Input	Port 1 7-bit input port P10 operates as an NMI input when a valid edge is input. Bit 0 of the P1 register indicates the NMI input status.	NMI
P11			INTP0
P12			INTP1
P13			INTP2
P14			INTP3/TI
P15			INTP4
P16			INTP5
P20	Input/output	Port 2 Input/output specifiable in bit units 6-bit input/output port	PWM
P21			TO00
P22			TO01
P23			TO20
P24			TO21
P25			TO30
P30		Port 3 Input/output specifiable in bit units 7-bit input/output port	TxD0/SB0/SO0
P31			RxD0/SB1/SI0
P32			TxC/$\overline{\text{SCK0}}$
P33			$\overline{\text{CTS0}}$
P34			TxD1/SO1
P35			RxD1/SI1
P36			$\overline{\text{CTS1}}$/$\overline{\text{SCK1}}$
P40-P47		Port 4 Input/output specifiable in bit units 8-bit input/output port	PD0 to PD7

Note Cannot be used as a general-purpose port (Non-maskable interrupt)

Pin Name	Input/Output	Function	Alternate Functions
P50	Input/output	Port 5 Input/output specifiable in bit units 3-bit input/output port	$\overline{\text{DATASTB}}$
P51			$\overline{\text{ACK}}$
P52			BUSY
P60-P63	Input	Port 6 Input/output specifiable in bit units 4-bit input port	ANI0 to ANI3
P70-P77	Input/output	Port 7 Input/output specifiable in bit units 8-bit input/output port	RTP0 to RTP7
P80		Port 8 Input/output specifiable in bit units 2-bit input/output port	DMARQ0
P81			DMARQ1

2.1.2 Non-port functions

(1) Bus control pins

Pin Name	Input/Output	Function	Alternate Functions
$\overline{\text{ASTB}}$	Output	External bus cycle address strobe signal output of external bus	—
$\overline{\text{RD}}$		External memory cycle data read strobe signal output of external bus	
$\overline{\text{WRL}}$		External memory cycle lower byte data write strobe signal output of external bus	
$\overline{\text{WRH}}$		External memory cycle upper byte data write strobe signal output of external bus	
READY	Input	External bus cycle ready signal input of external bus	
$\overline{\text{DEX}}$	Output	External bus cycle upper byte data enable signal output	
$\overline{\text{RAS}}$		DRAM row-address latch timing signal output	
D8/$\overline{\text{D16}}$	Input	External bus data bus width select signal input	
$\overline{\text{BUSLOCK}}$	Output	External bus lock signal output	
$\overline{\text{POLL}}$	Input	$\overline{\text{POLL}}$ signal (sampling during POLL instruction execution) input	
HLDRQ		External bus hold request signal input	
$\overline{\text{HLDAK}}$	Output	External bus hold acknowledge signal output	
$\overline{\text{REFRQ}}$		Refresh status signal output	
AD0 to AD15	3-state Input/output	External bus cycle address/data multiplex signal input/output of external bus	
A16 to A23	3-state output	External bus cycle address signal output of external bus	
$\overline{\text{IORD}}$	Output	External I/O cycle data read strobe signal output	
$\overline{\text{IOWR}}$		External I/O cycle data write strobe signal output	
DMARQ0	Input	DMA request signal input (channel 0)	P80
DMARQ1		DMA request signal input (channel 1)	P81
$\overline{\text{DMAAK0}}$	Output	DMA acknowledge signal output (channel 0)	—
$\overline{\text{DMAAK1}}$		DMA acknowledge signal output (channel 1)	
$\overline{\text{TCE0}}$		DMA end signal output (channel 0)	
$\overline{\text{TCE1}}$		DMA end signal output (channel 1)	

(2) Other pins

Pin Name	Input/Output	Function	Alternate Functions
GND	—	GND potential	—
V_{DD}		Positive power supply	
AV_{SS}		A/D converter GND potential	
AV_{DD}		A/D converter analog power supply	
AV_{REF}	Input	A/D converter reference voltage input	
$\overline{RESET}$		System reset signal input	
X1		System clock generation crystal resonator/ceramic resonator connection pin. For external clock supply, input it to the X1 pin and open the X2 pin.	
X2	—		
CLKOUT	Output	Internal system clock ϕ output	
$\overline{WDTOUT}$		Watchdog timer overflow signal output	
NMI	Input	Non-maskable interrupt request input[Note 1]	P10
INTP0		External interrupt request input[Note 2]	P11
INTP1			P12
INTP2			P13
INTP3			P14/TI
INTP4			P15
INTP5			P16
TI		External event clock input	P14/INTP3
PWM	Output	PWM output	P20
TO00, TO01, TO20, TO21, TO30		Timer unit output	P21 to P25
TxD0		UART transmit data output	P30/SB0/SO0
TxD1			P34/SO1
RxD0	Input	UART receive data input	P31/SB1/SI0
RxD1			P35/SI1
TxC	Output	UART transmit clock output	P32/$\overline{SCK0}$
$\overline{CTS0}$	Input	UART transmit enable signal input	P33
$\overline{CTS1}$			P36/$\overline{SCK1}$
SB0	Input/output	SBI transmit/receive data input/output	P30/TxD0/SO0
SB1			P31/RxD0/SI0

Notes 1. Since no NMI interrupt can be masked, an NMI interrupt is always started by valid edge detection (the pin level is read in a port 1 read operation.)

2. Can be used as a general-purpose input port by masking or disabling (IE = 0) each interrupt.

Pin Name	Input/Output	Function	Alternate Functions
SO0	Output	CSI transmit data output	P30/TxD0/SB0
SO1			P34/TxD1
SI0	Input	CSI receive data input	P31/RxD0/SB1
SI1			P35/RxD1
$\overline{SCK0}$	Input/output	CSI serial clock input/output	P32/TxC
$\overline{SCK1}$			P36/$\overline{CTS1}$
PD0 to PD7		Parallel interface data input/output	P40 to P47
$\overline{DATASTB}$		Parallel interface data strobe signal	P50
$\overline{ACK}$		Parallel interface acknowledge signal	P51
BUSY		Parallel interface busy signal	P52
ANI0 to ANI3	Input	Analog input signal to A/D converter	P60 to P63
RTP0 to RTP7	Output	Real-time output port	P70 to P77

2.1.3 Pin statuses

This section describes pin statuses after reset, in HOLD mode, and in the standby mode.
Alternate function (port/control) I/O pins are set to port mode after reset.

Table 2-1. Port Pin and Control Pin Statuses (1/2)

Pin Name	After Reset	When Control Mode is Specified		
		HOLD	Standby Mode	
			HALT	STOP
P00 to P07	Hi-Z	(No control mode specification) Hold for output pins, Hi-Z for input pins		
P10/NMI		Always Hi-Z because of input-only pin		
P11/INTP0				
P12/INTP1				
P13/INTP2				
P14/INTP3/TI				
P15/INTP4				
P16/INTP5				
P20/PWM		Continue		Hold
P21/TO00				
P22/TO01				
P23/TO20				
P24/TO21				
P25/TO30				
P30/TxD0/SB0/SO0		Type A		Type B
P31/RxD0/SB1/SI0		Type A		Type B
P32/TxC/$\overline{\text{SCK0}}$		Type A		Type B
P33/$\overline{\text{CTS0}}$		Hi-Z		Hi-Z
P34/TxD1/SO1		Continue		Hold
P35/RxD1/SI1		Hi-Z		Hi-Z
P36/$\overline{\text{CTS1}}$/$\overline{\text{SCK1}}$		Type A		Type B

Remark Continue : The specified operation continues.
Hold : The value just before switching to the mode is held.
Hi-Z : High-impedance
Type A : Hi-Z when the control signal input mode is selected. Continue when the control signal output mode is selected.
Type B : Hi-Z when the control signal input mode is selected. The last value is held when the control signal output mode is selected.

Table 2-1. Port Pin and Control Pin Statuses (2/2)

Pin Name	After Reset	When Control Mode is Specified		
		HOLD	Standby Mode	
			HALT	STOP
P40 to P47/PD0 to PD7	Hi-Z	Type A		Type B
P50/$\overline{\text{DATASTB}}$		Type A		Type B
P51/$\overline{\text{ACK}}$				
P52/BUSY				
P60 to P63/ANI0 to ANI3		Hi-Z		
P70 to P77/RTP0 to RTP7		Continue	Continue	Hold
P80/DMARQ0		Hi-Z		
P81/DMARQ1				

Remark Continue : The specified operation continues.
Hold : The value just before switching to the mode is held.
Hi-Z : High-impedance
Type A : Hi-Z when the control signal input mode is selected. Continue when the control signal output mode is selected.
Type B : Hi-Z when the control signal input mode is selected. The last value is held when the control signal output mode is selected.

Table 2-2. Non-Port Pin Statuses

Pin Name	After Reset	HOLD	HALT	STOP
$\overline{\text{RESET}}$	Hi-Z			
CLKOUT	Continue	Continue	Continue	Hold
$\overline{\text{ASTB}}$	Hi-Z	Hi-Z	Continue	H
$\overline{\text{RD}}$				
$\overline{\text{WRL}}$				
$\overline{\text{WRH}}$				
READY	Hi-Z			
$\overline{\text{DEX}}$	Hi-Z	Hi-Z	Continue	H
$\overline{\text{RAS}}$	Hi-Z	Hi-Z	Continue	H
$\overline{\text{REFRQ}}$	H		Continue	H
D8/$\overline{\text{D16}}$	Hi-Z			
$\overline{\text{BUSLOCK}}$	Hi-Z	Hi-Z	H	H
$\overline{\text{POLL}}$	Hi-Z			
HLDRQ	Hi-Z			
$\overline{\text{HLDAK}}$	Hi-Z	Continue	Continue	H
AD0 to AD15	Hi-Z	Hi-Z	Continue	Hold
A16 to A23	Hi-Z	Hi-Z	Continue	See Caution.
$\overline{\text{IORD}}$	Hi-Z	Hi-Z	H	H
$\overline{\text{IOWR}}$				
$\overline{\text{WDTOUT}}$	Continue	Continue	Continue	H
$\overline{\text{DMAAK0}}$	Hi-Z	H	Continue	Hold
$\overline{\text{DMAAK1}}$				
$\overline{\text{TCE0}}$				
$\overline{\text{TCE1}}$				

Remark Continue : The specified operation continues.
Hold : The value just before switching to the mode is held.
H : High level
Hi-Z : High-impedance

Caution Although address data is held, output addresses are undefined.

2.2 Description of Pin Functions

2.2.1 P00 to P07 (Port 0) ..... 3-state input/output

Port 1 is an 8-bit general-purpose input/output port. Pins P00 to P07 can be set as input or output ports bit-wise by port 0 mode register (PM0).

Table 2-3. Port 0 Operation (n = 0 to 7)

	Port Mode	
	PM0n = 1	PM0n = 0
P00	Input port	Output port
P01	Input port	Output port
P02	Input port	Output port
P03	Input port	Output port
P04	Input port	Output port
P05	Input port	Output port
P06	Input port	Output port
P07	Input port	Output port

2.2.2 P10 to P16 (Port 1) ..... input

Port 1 is a 7-bit special input pin which functions as port 1 and interrupt input pin. Besides serving as input-only port, P10 to P16 function as external interrupt input or capture trigger input pin or A/D converter trigger input pin for the timer unit.

Although this port also serves as interrupt input pin, the interrupt function and the input port function cannot be switched.

When it is desired to use only the input port function, mask the corresponding interrupt and capture operation (system reset automatically makes interrupts except NMI, masked and disables the capture operation).

Table 2-4. Port 1 Operation

	Function
P10	NMI input, or input-only port
P11	INTP0 input, or input-only port
P12	INTP1 input, or input-only port
P13	INTP2 input, or input-only port
P14	INTP3 input/TI input, or input-only port
P15	INTP4 input, or input-only port
P16	INTP5 input, or input-only port

(a) NMI (Non-Maskable Interrupt) ..... P10 alternate function input

An interrupt request input from this pin cannot be masked by software. Non-maskable interrupts are always acknowledgeable with the CPU and have priority over any other interrupts.

NMI input is edge-detected and the valid edge is specified with the interrupt mode register (INTM). Upon acknowledgment of NMI, vector No. 2 vectored interrupt is generated after termination of the instruction being executed. This NMI input is also used to clear the standby mode. The NMI input status can be monitored by reading P10.

(b) INTP0 to INTP5 (Interrupt from Peripheral 0 to 5) ..... P11 to P16, P14/TI alternate function inputs

An interrupt request input from these pins are maskable by software. INTPn (n = 0 to 5) input is edge-detected and the valid edge is specified with the interrupt mode register (INTM). In addition to serving as a latch timing input for the timer unit capture register, this INTPn input is also used to clear the HALT mode.

INTP5 can be used as an INIT signal input in the Centronics interface of a parallel interface function.

(c) TI (Timer Event Clock) ..... P14/INTP3 alternate function input

The TI pin is an external event clock input pin of timer 1.

2.2.3 P20 to P25 (Port 2) ... 3-state input/output

This is a 6-bit input/output alternate function pin which functions as port 2 and various control signals.

The port or control mode can be selected for the operating mode of P20 to P25 pins in bit units with port 2 mode control register (PMC2).

Table 2-5. Port 2 Operation (n = 0 to 5)

	Control Mode	Port Mode	
	PMC2n = 1	PMC2n = 0	
		PM2n = 1	PM2n = 0
P20	PWM output	Input port	Output port
P21	TO00 output	Input port	Output port
P22	TO01 output	Input port	Output port
P23	TO20 output	Input port	Output port
P24	TO21 output	Input port	Output port
P25	TO30 output	Input port	Output port

(1) Port mode

P20 to P25 pins can be set to the input or output port in bit units with port 2 mode register (PM2).

(2) Control mode

P20 to P25 pins can be set to the control mode in bit units with the PMC2 register (PMC2n = 1; n = 0 to 5).

(a) PWM (Pulse Width Modulation) ..... P20 alternate function output

This pin outputs a pulse signal from the PWM unit.

(b) TO00, TO01, TO20, TO21, TO30 (Timer Output) ..... P21 to P25 alternate function outputs

These pins are intended for output from each of three timers (TM0, TM2, TM3).

2.2.4 P30 to P36 (Port 3) ... 3-state input/output

Port 3 is a 7-bit input/output alternate function pin which functions as port 3 and various control signals.

The port or control mode can be selected for the operating mode of P30 to P36 pins in bit units with port 3 mode control register (PMC3).

Table 2-6. Port 3 Operation (n = 0 to 6)

	Control Mode	Port Mode	
	PMC3n = 1	PMC3n = 0	
		PM3n = 1	PM3n = 0
P30	TxD0 output/SB0 input-output/SO0 output	Input port	Output port
P31	RxD0 input/SB1 input-output/SI0 input	Input port	Output port
P32	TxC output/$\overline{\text{SCK0}}$ input-output	Input port	Output port
P33	$\overline{\text{CTS0}}$ input	Input port	Output port
P34	TxD1 output/SO1 output	Input port	Output port
P35	RxD1 input/SI1 input	Input port	Output port
P36	$\overline{\text{CTS1}}$ input/$\overline{\text{SCK1}}$ input-output	Input port	Output port

(1) Port mode

P30 to P36 pins can be set to the input or output port in bit units with port 3 mode register (PM3).

(2) Control mode

P30 to P36 pins can be set to the control mode in bit units with the PMC3 register (PMC3n = 1; n = 0 to 6). In the control mode, one of the alternate functions is selected when each serial interface protocol is selected.

(a) TxD0, TxD1 (Transmit Data 0, 1) ..... P30 and P34 alternate function outputs

These pins are ch0 and ch1 transmit data output pins of UART.

(b) RxD0, RxD1 (Receive Data 0, 1) ..... P31 and P35 alternate function inputs

These pins are ch0 and ch1 receive data input pins of UART.

(c) SB0, SB1 (Serial Bus 0, 1) ..... P30 and P31 alternate function inputs/outputs

These pins are ch0 and ch1 transmit/receive data input/output pins in the SBI mode.

(d) SI0, SI1 input (Serial Input 0, 1) ..... P31 and P35 alternate function inputs

These pins are ch0 and ch1 receive data input pins in the 3-wire serial I/O interface mode.

(e) SO0, SO1 output (Serial Output 0, 1) ..... P30 and P34 alternate function outputs

These pins are ch0 and ch1 transmit data output pins in the 3-wire serial I/O interface mode.

(f) TxC (Transmit Clock) ..... P32 alternate function output

This is a UART transmit clock output pin.

(g) $\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$ (Serial Clock 0, 1) ..... P32 and P36 alternate function inputs/outputs

These pins are ch0 and ch1 serial clock input/output pins of the clocked serial interface (SBI mode, 3-wire I/O interface mode).

(h) $\overline{\text{CTS0}}$, $\overline{\text{CTS1}}$ (Clear to Send 0, 1) ..... P33 and P36 alternate function inputs

These are UART ch0 and ch1 CTS pins. They are input pins to enable transmission.

2.2.5 P40 to P47 (Port 4) ..... 3-state input/output

This is an 8-bit input/output alternate function pin which functions as port 4 and parallel interface at input/output pin.

The port or control mode can be selected for the operating mode of P40 to P47 pins in 8-bit units with port 4 mode control register (PMC4).

Table 2-7. Port 4 Operation (n = 0 to 7)

	Control Mode	Port Mode	
	$PMC4_7$ = 1	$PMC4_7$ = 0	
		PM4n = 1	PM4n = 0
P40	PD0 input/output	Input port	Output port
P41	PD1 input/output	Input port	Output port
P42	PD2 input/output	Input port	Output port
P43	PD3 input/output	Input port	Output port
P44	PD4 input/output	Input port	Output port
P45	PD5 input/output	Input port	Output port
P46	PD6 input/output	Input port	Output port
P47	PD7 input/output	Input port	Output port

(1) Port mode

P40 to P47 pins can be set to the input or output port in bit units with port 4 mode register (PM4).

(2) Control mode

P40 to P47 pins can be set to the control mode in 8-bit units with the PMC4 register ($PMC4_7$ = 1).

(a) PD0 to PD7 (Parallel Data 0 to 7) ..... P40 to P47 alternate function input/output

These are parallel interface transmit/receive data input/output pins.

2.2.6 P50 to P52 (Port 5) ..... 3-state input/output

Port 5 is a 3-bit input/output alternate function pin which functions as port 5 and parallel interface control signal.

The port or control mode can be selected for the operating mode of P50 to P52 pins in bit units with port 5 mode control register (PMC5).

Table 2-8. Port 5 Operation (n = 0 to 2)

	Control Mode	Port Mode	
	PMC5n = 1	PMC5n = 0	
		PM5n = 1	PM5n = 0
P50	$\overline{\text{DATASTB}}$ input/output	Input port	Output port
P51	$\overline{\text{ACK}}$ input/output	Input port	Output port
P52	BUSY input/output	Input port	Output port

(1) Port mode

P50 to P52 pins can be set to the input or output port in bit units with port 5 mode register (PM5).

(2) Control mode

P50 to P52 pins can be set to the control mode in bit units with the PMC5 register (PMC5n = 1; n = 0 to 2).

(a) $\overline{\text{DATASTB}}$ (Data Strobe) ..... P50 alternate function input/output

The $\overline{\text{DATASTB}}$ is a strobe signal input/output pin in the parallel interface function.

In input mode:

This pin serves as a strobe signal input pin for writing receive data in the parallel interface buffer (PAB). Data of PD0 to PD7 are fetched to the PAB register in synchronization with the rising or falling edge of this signal, and the BUSY signal becomes high. The edge is specified according to the setting of the parallel interface control register 0 (PAC0).

In output mode:

This pin serves as a strobe signal output pin for latching transmit data with the external buffer register, etc.

(b) $\overline{\text{ACK}}$ (Acknowledge) ..... P51 alternate function input/output

The $\overline{\text{ACK}}$ is an input/output pin for an acknowledged signal which indicates transmission/reception completion with the PAB register in the parallel interface function.

In input mode:

This pin outputs an acknowledge signal which informs externals that the receive data in the PAB register has read out by DMA or instructions.

Three types of output timings can be selected according to the timings with the BUSY signal.

In output mode:

This pin inputs an acknowledge signal which indicates reception completion.

$\overline{\text{ACK}}$ pin = Detects the change of "high level → low level".

(c) BUSY (Busy) ..... P52 alternate function input/output

This pin serves as an input/output pin for the BUSY signal which indicates that transmission/reception is disable in the parallel interface function.

In input mode:

This pin outputs the BUSY signal which indicates that transmission/reception is disable.

The set/reset conditions of the pin level in the input mode are described below.

Set condition: Pin level goes high when one of the following conditions is met.

- $\overline{\text{DATASTB}}$ signal input (rising or falling synchronization)
- When the MB0 and MB1 bits of the parallel interface control register (PAC1) are set to "1×" by software.
- When the external interrupt INTP5 is input in case where the INIT bit of PAC1 register = 1.

Reset condition: Pin level goes low when all the following conditions are met.

- A certain period of time has elapsed (depending on the mode) after a receive data is read out from the PAB register (status register: IBF = 0).
- When the MB0 and MB1 bits of the PAC1 register are set to "00" by software.

In output mode:

This pin inputs the BUSY signal which indicates that transmission/reception is disable.

This pin detects the change of "high-level → low-level".

2.2.7 P60 to P63 (Port 6) ..... 3-state input

Port 6 is a 4-bit input alternate function pin which functions as port 6 and A/D converter analog input pin.

Although this port also serves as analog input pin, the input port and the analog input pin cannot be switched.

Table 2-9. Port 6 Operation

	Function
P60	ANI0 input or input-only port
P61	ANI1 input or input-only port
P62	ANI2 input or input-only port
P63	ANI3 input or input-only port

2.2.8 P70 to P77 (Port 7) ..... 3-state input/output

Port 7 is an 8-bit input/output alternate function pin which functions as port 7 and real-time output port.

The port or control mode can be selected for the operating mode of P70 to P77 pins in bit units with port 7 mode control register (PMC7).

Table 2-10. Port 7 Operation (n = 0 to 7)

	Control Mode	Port Mode	
	PMC7n = 1	PMC7n = 0	
		PM7n = 1	PM7n = 0
P70	RTP0 output	Input port	Output port
P71	RTP1 output	Input port	Output port
P72	RTP2 output	Input port	Output port
P73	RTP3 output	Input port	Output port
P74	RTP4 output	Input port	Output port
P75	RTP5 output	Input port	Output port
P76	RTP6 output	Input port	Output port
P77	RTP7 output	Input port	Output port

(1) Port mode

P70 to P77 pins can be set to the input or output port in bit units with port 7 mode register (PM7).

(2) Control mode

P70 to P77 pins can be set to the control mode in bit units with the PMC7 register (PMC7n = 1; n = 0 to 7).

(a) RTP0 to RTP7 (Real-time Output Port 0 to 7) ..... P70 to P77 alternate function outputs

This is a real-time output port output pin. RTP0 to RTP3 correspond to the P7L register which is the real-time output port buffer and RTP4 to RTP7 correspond to the P7H register and output in 4-bit units is enabled.

2.2.9 P80, P81 (Port 8) ..... 3-state input/output

Port 8 is a 2-bit input/output alternate function pin which functions as port 8 and DMA request signal input pin.

The port or control mode can be selected for the operating mode of P80 and P81 pins in bit units with port 8 mode control register (PMC8).

Table 2-11. Port 8 Operation (n = 0, 1)

	Control Mode	Port Mode	
	PMC8n = 1	PMC8n = 0	
		PM8n = 1	PM8n = 0
P80	DMARQ0 input	Input port	Output port
P81	DMARQ1 input	Input port	Output port

(1) Port mode

P80 and P81 pins can be set to the input or output port in bit units with port 8 mode register (PM8).

(2) Control mode

P80 and P81 pins can be set to the control mode in bit units with the PMC8 register (PMC8n = 1; n = 0, 1).

(a) DMARQ0, DMARQ1 (DMA Request 0, 1) ..... P80 and P81 alternate function inputs

These are external DMA request input pins of DMA controller (channel 0, 1).

It is sensed by the level in the demand release mode. In other transfer modes, it is triggered at the rising edge.

2.2.10 GND (Ground)

Fix all GND pins to the GND potential.

2.2.11 V_{DD} (Power Supply)

Positive power supply pin. Connect all V_{DD} pins to V_{DD}.

2.2.12 AV_{SS} (Analog V_{SS})

GND pin of A/D converter. Fix this pin to the GND potential.

2.2.13 AV_{DD} (Analog V_{DD})

Positive power supply pin of A/D converter

2.2.14 AV_{REF} (Analog Reference Voltage)

Reference voltage input pin of A/D converter

2.2.15 $\overline{\text{RESET}}$ (Reset) ..... input

$\overline{\text{RESET}}$ input is an asynchronous input. If a signal having the fixed low-level width is input irrespective of the operating clock, system reset is applied with priority over any other operation.

Besides normal initialize/start, this pin is also used to clear the standby (HALT/STOP) mode.

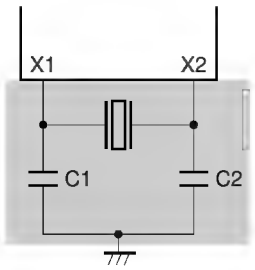
$\overline{\text{RESET}}$ is an active-low input pin.

2.2.16 X1, X2 (Crystal)

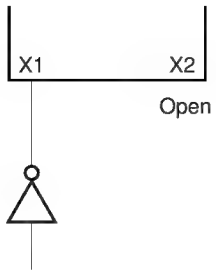
These are connect pins to connect crystal/ceramic resonators for system clock generation.

For external clock input, input it to the X1 pin and open the X2 pin.

(a) Crystal/ceramic[Note] oscillation



(b) External clock input



Cautions 1. Place the oscillator as near as possible the X1 and X2 pins.

2. Do not put any other signal conductor through the shaded area.

Note As for crystal resonator/ceramic resonator and external capacitance, take matching with the device into account.

2.2.17 CLKOUT (Clock Out) ..... output

This is an output pin of internal system clock to supply various clocks to the CPU and the peripheral hardware with the frequency set in the clock generator.

2.2.18 $\overline{\text{ASTB}}$ (Address Strobe) ..... output

The $\overline{\text{ASTB}}$ pin is used to output a strobe signal for external latch of the external bus address information.

2.2.19 $\overline{\text{RD}}$ (Memory Read Strobe) ..... output

The $\overline{\text{RD}}$ pin is a read strobe signal output pin which becomes active in the read cycle from the memory connected to the external bus.

2.2.20 $\overline{\text{WRL}}$ (Memory Write Low Strobe) ..... output

The $\overline{\text{WRL}}$ pin is used to output a lower 8-bit data write strobe signal which becomes active in the write cycle to the memory connected to the external bus.

This signal becomes active in the following bus cycles.

(1) Using 16-bit external data bus width

- Bus cycle with byte data access to an even address or second byte access to an odd address for word data.
- Bus cycle with word data access to an even address.

(2) Using 8-bit external data bus width

- Cycle of all external memory writes

2.2.21 $\overline{\text{WRH}}$ (Memory Write High Strobe) ..... output

The $\overline{\text{WRH}}$ pin is used to output a higher 8-bit data write strobe signal which becomes active in the write cycle to the memory connected to the external bus.

This is valid only when the external data bus width is 16 bits (always inactive when 8 bits.)

Table 2-12. $\overline{WRL}$ and $\overline{WRH}$ Pins Operation in the Memory Write Cycle (Using 16-bit External Data Bus Width)

		$\overline{WRL}$	$\overline{WRH}$
Byte Access	Even Address	Active	Inactive
	Odd Address	Inactive	Active
Word Access	Even Address	Active	Active
	Odd[Note] Address	Active only at the 2nd bus cycle	Active only at the 1st bus cycle

Note Activates two bus cycles.

2.2.22 READY (Ready) ..... input

This is an input pin to control external insertion of a wait state in the external bus cycle.

The READY pin is sensed according to the level and a wait state is inserted at the low level.

Caution Valid only when one or more waits are set by the PWC register.

2.2.23 $\overline{DEX}$ (Data Exchange) ..... output

This is a signal pin to indicate an access to the higher 8 bits (D15 to D8) of the external data bus. It changes at the falling edge of the T1 cycle CLKOUT signal. The lower 8 bits (D7 to D0) of the external data bus are controlled with A0.

Access		$\overline{DEX}$	A0
Even address word		0	0
Odd address word	1st time	0	1
	2nd time	1	0
Even address byte		1	0
Odd address byte		0	1

Caution The $\overline{DEX}$ pin becomes undefined when it has an 8-bit bus width. In that case, do not connect anything to the $\overline{DEX}$ pin.

2.2.24 $\overline{RAS}$ (Row Address Strobe) ..... output

This is a pin to output row-address latch timings to the DRAM connected to the external bus.

2.2.25 D8/$\overline{D16}$ (Data 8/16) ..... input

This is an input pin to determine whether the external data bus width should be 8 or 16 bits.

This is used, fixed either at the low level or high level.

2.2.26 $\overline{BUSLOCK}$ (Buslock) ..... output

This is a pin to indicate "bus use prohibited" to bus users except the CPU (for example, the DMA controller or other processors). This pin becomes active while one instruction following the BUSLOCK prefix instruction is being executed.

During the bus lock period in which this signal remains active, no bus request (including DMA transfer request, refresh request) other than that generated from the CPU to the bus can be acknowledged.

2.2.27 $\overline{POLL}$ (Poll) ..... input

The $\overline{POLL}$ input signal is checked with the POLL instruction. If it is at the low level, execution moves to the next instruction. If at the high level, the $\overline{POLL}$ input is checked every five clock cycles until it is set to the low level.

This function is used to synchronize the CPU program with an external device.

2.2.28 HLDRQ (Hold Request) ..... input

This is a signal input pin for an external device to request the V55PI for the release of external buses (address bus, data bus, and control bus).

HLDRQ is an active-high signal input pin.

2.2.29 $\overline{HLDAK}$ (Hold Acknowledge) ..... output

This is an acknowledge signal output pin to indicate that the CPU has made the external bus high-impedance upon receipt of the hold request signal (HLDRQ).

While this signal is active (at low level), the external buses (address bus, data bus, and control bus) remains high-impedance.

2.2.30 AD0 to AD15 (Address/Data 0 to 15) ..... input/output

This is a multiplex signal input/output pin of external bus address/data.

2.2.31 A16 to A23 (Address 16 to 23) ..... output

This is an external bus address signal output pin.

2.2.32 $\overline{IORD}$ (I/O Read) ..... output

This is a read strobe signal output pin which becomes active in the external I/O read cycle.

This signal is activated not only in an I/O read cycle by the CPU but also in a DMA transfer from the external I/O.

$\overline{IORD}$ is an active-low output pin.

2.2.33 $\overline{\text{IOWR}}$ (I/O Write) ..... output

This is a write strobe signal output pin which becomes active in the external I/O write cycle.

This signal is activated not only in an I/O write cycle by the CPU but also in a DMA transfer to the external I/O.

$\overline{\text{IOWR}}$ is an active-low output pin.

2.2.34 $\overline{\text{WDTOUT}}$ (Watchdog Timer Output) ..... output

This is a pin to output a signal which indicates that the watchdog timer has generated a non-maskable interrupt.

2.2.35 $\overline{\text{DMAAK0}}$, $\overline{\text{DMAAK1}}$ (DMA Acknowledge 0, 1) ..... output

These are DMA acknowledge signal output pins of DMA controller (channels 0 and 1). This signal is output to the external DMA request DMARQ0 and DMARQ1 and is also output when the DMA transfer between I/O and memory is executed by the internal timer. The DMA acknowledge signal is not output in DMA transfer between memories, memory and SFR (special-function register). They are active-low output pins.

2.2.36 $\overline{\text{TCE0}}$, $\overline{\text{TCE1}}$ (Terminal Count End 0, 1) ..... output

These are pins to output the DMA end signal of DMA controller (channels 0 and 1).

When an external I/O is specified as a transfer target, they are valid in the intelligent DMA mode-2, 2-channel operation mode (stop at end), and memory-to-memory transfer mode (stop at end). This signal is output when a borrow is generated by the down counting of the terminal counter (TC0 and TC1) in each mode. In other transfer targets and operation modes, these pins are always inactive. They are active-low output pins.

2.2.37 $\overline{\text{REFRQ}}$ (Refresh Request) ..... output

This pin becomes active in the refresh cycle to inform externals of the refresh cycle.

$\overline{\text{REFRQ}}$ is an active-low output pin.

2.3 Pin Input/Output Circuits

2.3.1 Input/output circuit types

(1) Ports

Pin Name	Type	Pin Name	Type
P00 to P07	5	P32/TxC/$\overline{SCK0}$	2
P10/NMI	2	P33/$\overline{CTS0}$	
P11/INTP0		P34/TxD1/SO1	5
P12/INTP1		P35/RxD1/SI1	2
P13/INTP2		P36/$\overline{CTS1}$/$\overline{SCK1}$	
P14/INTP3/TI		P40 to P47/PD0 to PD7	5
P15/INTP4		P50/$\overline{DATASTB}$	
P16/INTP5		P51/$\overline{ACK}$	
P20/PWM	5	P52/BUSY	
P21/TO00		P60 to P63/ANI0 to ANI3	9
P22/TO01		P70 to P77/RTP0 to RTP7	5
P23/TO20		P80/DMARQ0	
P24/TO21		P81/DMARQ1	
P25/TO30			
P30/TxD0/SB0/SO0	2		
P31/RxD0/SB1/SI0			

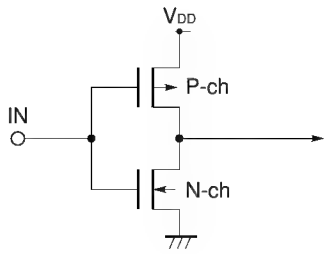
(2) Non-ports

Pin Name	Type	Pin Name	Type
$\overline{RESET}$	2	AD0 to AD15	5
READY	1	A16 to A23	4
CLKOUT	3	$\overline{RAS}$	
$\overline{ASTB}$	4	$\overline{IORD}$	
$\overline{RD}$		$\overline{IOWR}$	
$\overline{WRL}$		D8/$\overline{D16}$	1
$\overline{WRH}$		$\overline{DEX}$	4
$\overline{BUSLOCK}$		$\overline{DMAAK0}$	
$\overline{HLDAK}$		$\overline{DMAAK1}$	
$\overline{WDTOUT}$		$\overline{TCE0}$	
HLDRQ	1	$\overline{TCE1}$	
$\overline{POLL}$		$\overline{REFRQ}$	

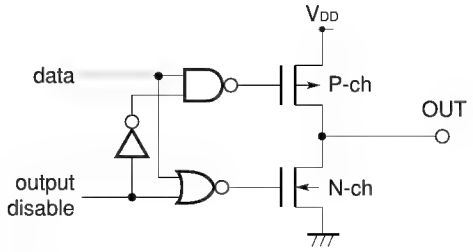
2.3.2 Input/output circuits

Figure 2-1. Input/Output Circuits Classified by Types

Type 1

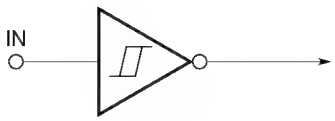


Type 4



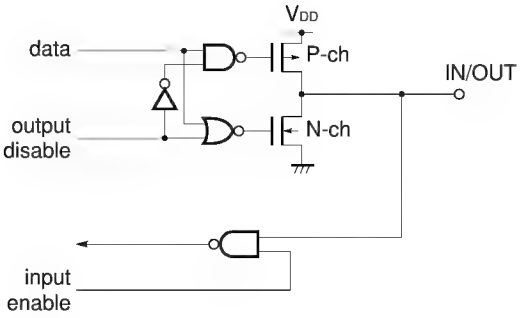
Push-pull output which can be made output high-impedance (with both P-ch and N-ch set to OFF)

Type 2

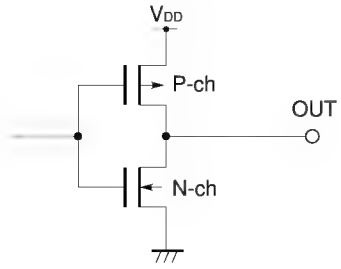


Schmitt trigger input having hysteresis characteristics

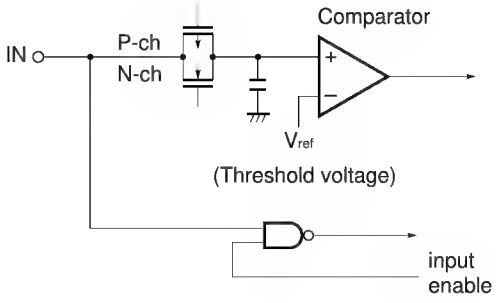
Type 5



Type 3



Type 9



2.4 Recommended Connections of Unused Pins

The following are the recommended connections of unused pins.
When connecting to V_{DD} or GND via a resistor, it is recommended that a resistor of 1 kΩ to 10 kΩ should be used.

(1) Ports

Pin Name	Recommended Connecting Method
P00 to P07	Input status : Connect to V_{DD} via a resistor respectively. Output status : Open
P10/NMI	Connect to GND via a resistor.
P11/INTP0	Connect to V_{DD} or GND via a resistor.
P12/INTP1	
P13/INTP2	
P14/INTP3/TI	
P15/INTP4	
P16/INTP5	
P20/PWM	Input status : Connect to V_{DD} via a resistor respectively. Output status : Open
P21/TO00	
P22/TO01	
P23/TO20	
P24/TO21	
P25/TO30	
P30/TxD0/SB0/SO0	Input status : Connect to V_{DD} via a resistor respectively. Output status : Open
P31/RxD0/SB1/SI0	
P32/TxC/$\overline{SCK0}$	
P33/$\overline{CTS0}$	
P34/TxD1/SO1	
P35/RxD1/SI1	
P36/$\overline{CTS1}$/$\overline{SCK1}$	
P40 to P47/PD0 to PD7	Input status : Connect to V_{DD} via a resistor respectively. Output status : Open
P50/$\overline{DATASTB}$	
P51/$\overline{ACK}$	
P52/BUSY	
★ P60 to P63/ANI0 to ANI3	Connect to GND via a resistor.
P70 to P77/RTP0 to RTP7	Input status : Connect to V_{DD} via a resistor respectively. Output status : Open
P80/DMARQ0	
P81/DMARQ1	

Remark Each port pin becomes an input port after reset is released.

(2) Non-ports

Pin Name	Recommended Connecting Method
CLKOUT	Open
$\overline{\text{ASTB}}$	
$\overline{\text{RD}}$	
$\overline{\text{WRH}}$	
$\overline{\text{WRL}}$	
A16 to A23	
$\overline{\text{DEX}}$	
$\overline{\text{RAS}}$	
$\overline{\text{HLDAK}}$	
$\overline{\text{BUSLOCK}}$	
$\overline{\text{WDTOUT}}$	
$\overline{\text{DMAAK0}}$	
$\overline{\text{DMAAK1}}$	
$\overline{\text{TCE0}}$	
$\overline{\text{TCE1}}$	
$\overline{\text{IORD}}$	
$\overline{\text{IOWR}}$	
$\overline{\text{REFRQ}}$	
HLDRQ	Connect to GND via a resistor.
$\overline{\text{POLL}}$	
READY	Connect to V_{DD} via a resistor.
AD0 to AD15	Connect to V_{DD} via a resistor respectively.
AV_{SS}	Connect to GND via a resistor.
AV_{REF}	
AV_{DD}	Connect to V_{DD} via a resistor.

(3) Others

Pin Name	Recommended Connecting Method
IC (H)	Connect to V_{DD} via resistor.
IC (L)	Connect to GND via resistor.
OPEN	Open

CHAPTER 3 CPU FUNCTIONS

The V55PI has a CPU which is software upward compatible with V20, V30 (native mode), V25 and V35 in object code level.

3.1 Features

- Software upward compatible with V20, V30 (native mode), V25 and V35 (additional instructions available)
- Minimum instruction cycle: 160 ns/12.5 MHz (when externally operated at 25 MHz)
  125 ns/16 MHz (when externally operated at 32 MHz)
- Address space: 16 Mbytes — Basic memory (program/data/stack) space 1 Mbyte
  Expansion memory (data) space 16 Mbytes
- Register file space (internal RAM): 512 bytes/16 register banks
- I/O space: 64 Kbytes
- Register configuration (comparison with V20 and V30, V25 and V35)

Item		V20, V30	V25, V35	**V55PI**
Expansion segment register		None	None	DS2, DS3
Register bank		None	8 Banks (Memory space)	16 Banks (Register file space)
PSW	Mode flag	MD	None	None
	Register bank flag	None	RB0 to RB2	RB0 to RB3
	Input/output instruction trap flag	None	$\overline{\text{IBRK}}$	$\overline{\text{IBRK}}$
	User flag	None	F0, F1	None
Special function register area		None	240 bytes (Memory mapped in FFF00H to FFFEFH)	496 bytes (Memory mapped in FFE00H to FFFEFH)

- 16-bit internal architecture: 16-/8-bit external data bus width selection enabled
- Automatic wait control by dividing memory space into variable size (6 blocks max.)
  – Programmable wait function
  – Wait function with READY pin
- Refresh function
  Automatically generates refresh cycle ($\overline{\text{REFRQ}}$ output, $\overline{\text{RAS}}$ output)

3.2 Registers

The CPU of V55PI has a general register set compatible with V20, V30 (native mode), V25 and V35. The general register set is mapped in the register file space and also serves as the internal RAM. Maximum of 16 register sets can be located in bank format in the internal RAM.

Further, various special-function registers are used to control the on-chip peripheral hardware. They are mapped in the memory space 0FFE00H to 0FFFEFH.

3.2.1 Register banks

The general register set is mapped in the register file space (internal RAM). The general register set is in bank format and up to 16 banks, each having 32 bytes, can be set.

Normally, the CPU executes a program using register bank 15 and can automatically switch it to another register bank with a maskable hardware interrupt or a software interrupt (BRKCS instruction). The return instruction (RETRBI) from the interrupt can be used to return from the switched register bank to the original register bank.

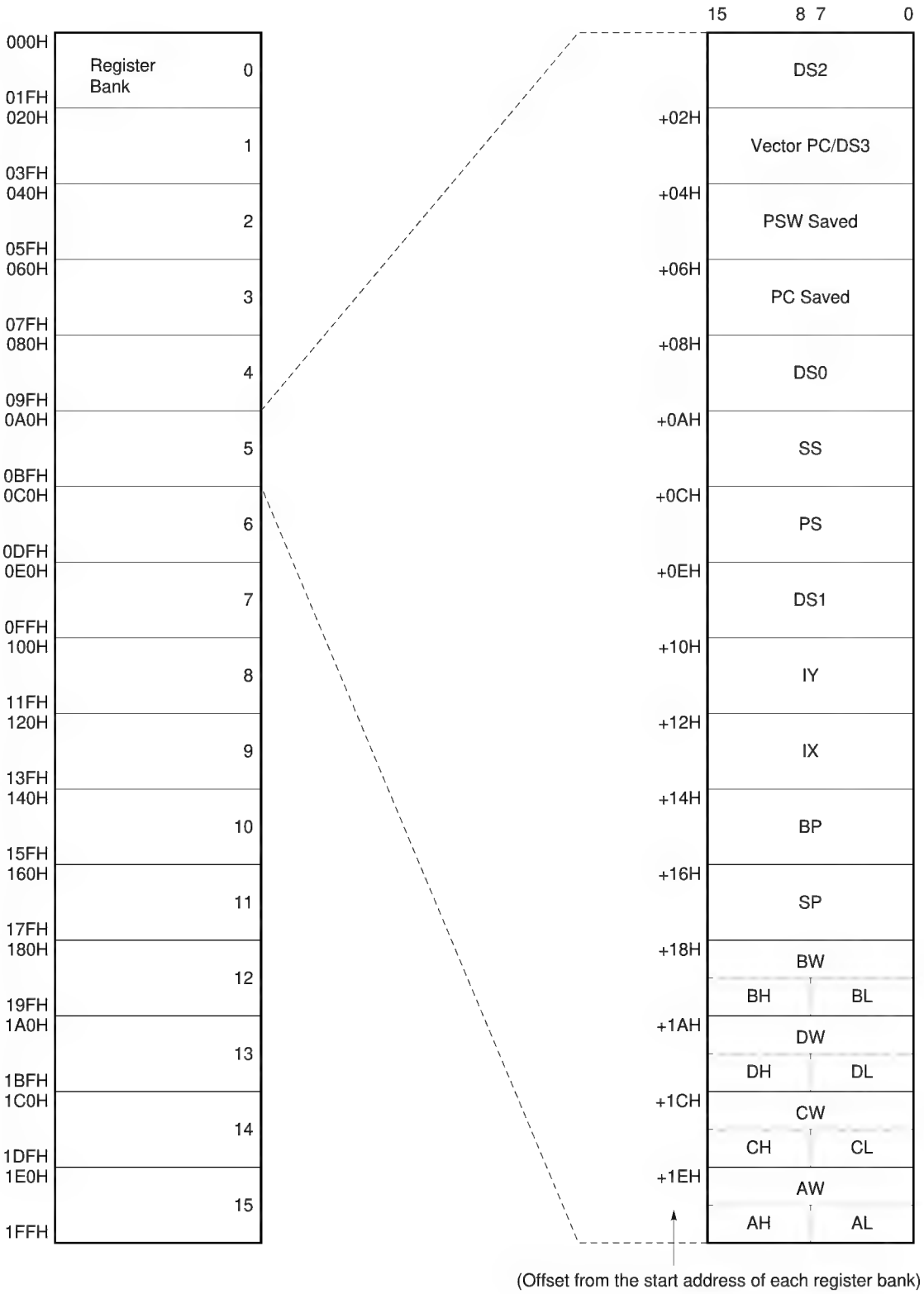
Figure 3-1 shows the register bank configuration. The general register set is mapped in the (+08H) to (+1FH) area by offset from the start address of each register bank. The first word area of the register bank is the expansion segment register (DS2) area. The vector PC/DS3 area is intended to set the start address offset value of the interrupt service routine. This area is also used as the expansion segment register (DS3) area. The PSW save area is intended to save PSW when the register bank is switched. The PC save area is intended to save PC when the register bank is switched.

After a reset, register bank 15 is automatically selected. Segment register initialization after a reset is executed on register bank 15 only.

The register file space where the general register set has been mapped can also be accessed as data memory by adding the dedicated prefix instruction (IRAM:) to the memory manipulation instruction.

Among other 16 sets of the register banks, banks 0 and 1 are allocated a macro service channel (parameter and work area for macro service) in overlapped form.

Figure 3-1. Register Bank Configuration



3.2.2 General registers (AW, BW, CW, and DW)

Four 16-bit registers are used as general registers. These registers are accessible not only as 16-bit registers but also as 8-bit registers (AH, AL, BH, BL, CH, CL, DH, and DL) while each register is divided into higher 8 bits and lower 8 bits.

These registers are used as 8-bit registers or 16-bit registers in the wide range of instructions such as transfer instruction, arithmetic instruction, or logical operation instruction.

Each register is used as default register for the following particular instruction processing.

AW : Word multiplication/division, word input/output, and data conversion
AL : Byte multiplication/division, byte input/output, BCD rotation, and data conversion
AH : Byte multiplication/division
BW : Data conversion
CW : Loop control branch and repeat prefix
CL : Shift instruction, rotate instruction, and BCD operation
DW : Word multiplication/division and indirect addressing input/output

These registers are mapped in the register file space (internal RAM). The address of each register is one obtained by adding each offset value to (register bank number × 32).

Table 3-1. General Register Offset

Register	Offset	Register	Offset
AW	1EH	AL	1EH
		AH	1FH
BW	18H	BL	18H
		BH	19H
CW	1CH	CL	1CH
		CH	1DH
DW	1AH	DL	1AH
		DH	1BH

3.2.3 Pointers (SP and BP) and index registers (IX and IY)

These are 16-bit registers to be used as base pointer or index register for memory access by based addressing (BP), indexed addressing (IX and IY), or based indexed addressing (BP, IX, and IY). SP is used as pointer for stack manipulation. Like general registers, these registers are used for instructions such as transfer or arithmetic operation. In that case, they cannot be used as 8-bit registers. Each register is used as fixed address pointer for the following particular processing.

SP : Stack manipulation
IX : Primitive block transfer instruction and BCD operation instruction source side address specification
IY : Primitive block transfer instruction and BCD operation instruction destination side address specification

These registers are mapped in the register file space which also serves as the internal RAM. The address of each register is one obtained by adding each offset value to (register bank number × 32).

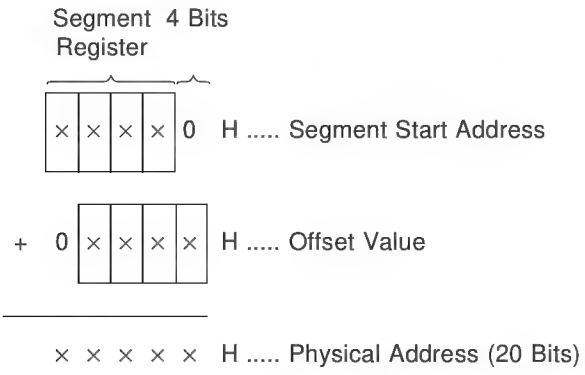
Table 3-2. Pointer and Index Register Offset

Register	Offset
SP	16H
BP	14H
IX	12H
IY	10H

3.2.4 Segment registers (PS, SS, DS0, and DS1)

The CPU controls the 1-Mbyte basic memory space by dividing it into 64-Kbyte unit logic segments. The CPU specifies the start address of each segment with the segment register and specifies the relative address from the start address as offset with another register or effective address.

A physical address is created as follows.



There are four types of segment registers, PS (Program Segment), SS (Stack Segment), DS0 (Data Segment0), and DS1 (Data Segment1). Each segment register is used for the following operations.

PS : Program code fetch.
SS : Stack manipulate instruction and addressing using BP as base register.
DS0 : Access for general variable and access for source block data by primitive block transfer instruction and BCD operation instruction.
DS1 : Access for destination block data by primitive block transfer instruction and BCD operation instruction.

If the segment override prefix is used, general variable access can be made from a segment register other than DS0. In addressing using BP as base register, a segment register other than SS can also be used.

Example

```
MOV  AW,  1000H
MOV  DS1, AW
MOV  BL, DS1 : BYTE PTR [IX]    ; Byte data read from DS1 : IX
```

After a reset, PS of register bank 15 is initialized to FFFFH, and SS, DS0 and DS1 are initialized to 0000H.

These registers are mapped in the register file space (internal RAM). The address of each register is one obtained by adding each offset value to (register bank number × 32).

Table 3-3. Segment Register Offset

Register	Offset
DS0	08H
DS1	0EH
SS	0AH
PS	0CH

3.2.5 Expansion segment registers (DS2, DS3)

In addition to the segment registers to access the 1M-byte basic memory space, the V55PI is provided with expansion segment registers to divide the 16M-byte expansion memory space in 64K-byte segments and to specify the start address of each segment. Expansion segment registers, DS2 (Data Segment2) and DS3 (Data Segment3), are used as follows.

DS2 : Access for general variable in the expansion memory space (by segment override prefix) and access for source block data by primitive block transfer instruction and BCD operation instruction to the expansion memory space.

DS3 : Access for general variable in the expansion memory space (by segment override prefix) and access for destination block data by primitive block transfer instruction and BCD operation instruction to the expansion memory space.

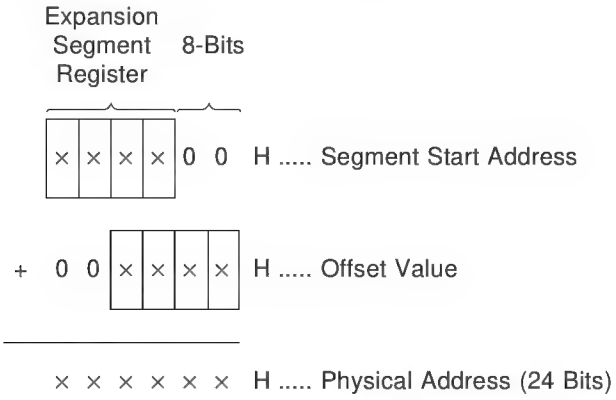
A data access using the expansion segment register is performed by using the segment override prefix. Also, especially in the primitive block transfer instruction and BCD operation instruction, DS2 and DS3 are doubly specified with the segment override prefix (in this case, the specification order of DS2 and DS3 is arbitrary.)

Example

```
REP
DS2:
DS3:   MOVBKW ; Word memory block transfer from DS2 : IX to DS3 : IY
```

An access is made by specifying the start address of each segment with the expansion segment register and specifying the relative address from the start address as offset with another register or immediate data.

A physical address is created as follows.



After a reset, DS2 and DS3 of register bank 15 are initialized to 0000H.

These registers are mapped in the register file space (internal RAM). The address of each register is one obtained by adding each offset value to (register bank number × 32).

Table 3-4. Expansion Segment Register Offset

Register	Offset
DS2	00H
DS3	02H (This register also serves as vector PC)

3.2.6 Special-function registers (SFR)

The V55PI has a group of registers with a function to control the on-chip peripheral hardware.

For each peripheral hardware, several special-function registers are available depending on what to be controlled. Actual operations can be set using each bit in the register. These registers are mapped in the memory space and are read/written using the same procedure as that of the normal memory (refer to **3.5.3 Special-function register area**).

Example

```
MOV  AW,  0FFE0H
MOV  DS1, AW
MOV  BL, DS1 : BYTE PTR [1EFH]    ; 0FFE0H : 1EFH (PRC register) is read
```

The BTCLR and BTCLRL instructions are only valid for the special-function registers. The BTCLRL instruction has been added for use with the V25 and V35.

The BTCLR instruction is valid for registers in the higher 240 bytes (0FFF00H to 0FFFEFH) in the special-function register area and the BTCLRL instruction is valid for registers in the lower 256 bytes (0FFE00H to 0FFEFFH).

3.3 Program Counter (PC)

This is a 16-bit binary counter to hold the offset value of a program memory address which the CPU is ready to execute.

The PC is incremented each time an instruction code is fetched from the instruction queue. A new location address value is loaded when the branch, call, return, or break instruction is executed.

After a reset, 0000H is loaded to the PC. The PS is initialized to FFFFH after a reset. Thus, the CPU starts execution at physical address 0FFFF0H after a reset.

3.4 Program Status Word (PSW)

The PSW consists of six status flags and five control flags.

- Status flags
  - V (Overflow) ... Overflow detection flag
  - S (Sign) ... Sign bit detection flag
  - Z (Zero) ... All zero detection flag
  - AC (Auxiliary Carry) ... Carry and borrow detection flag in 4-bit units
  - P (Parity) ... Parity detection flag
  - CY (Carry) ... Carry and borrow detection flag
- Control flags
  - RB0 to RB3 (Register Bank 0 to 3) ... Register bank directive flag
  - DIR (Direction) ... Block transfer/input/output instruction directions control flag
  - IE (Interrupt Enable) ... Interrupt enable status control flag
  - BRK (Break) ... Single step interrupt control flag
  - $\overline{\text{IBRK}}$ (I/O Break) ... Input/output instruction trap control flag

The status flags are automatically set to (1) or reset to (0) in accordance with the result (data value) of each instruction execution. The CY flag can be directly set/reset/reversed with an instruction.

The control flags are set/reset with an instruction and control the CPU operation. The IE and BRK flags are reset upon start of interrupt service.

PSW contents can be saved/returned to the stack with the PUSH/POP instruction. However, bits 12 to 15 (RB0 to RB3) are not returned to PSW with the POP PSW instruction.

The lower 8 bits of PSW can be saved/returned to the AH register with the MOV instruction.

Figure 3-2. PSW Bit Configuration

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RB3	RB2	RB1	RB0	V	DIR	IE	BRK	S	Z	0	AC	0	P	$\overline{\text{IBRK}}$	CY

3.4.1 CY (Carry Flag)

(1) Binary add/subtract

In the case of byte operation, CY is set if a carry or a borrow is generated out of bit 7 of the operation result. If none of them is generated, this flag is reset.

In the case of word operation, CY is set if a carry or a borrow is generated out of bit 15 of the operation result. If none of them is generated, this flag is reset.

CY remains unchanged with the increment and decrement instructions.

(2) Logical operation

CY is reset regardless of the logical operation result.

(3) Binary multiply

If AH is 0 as a result of unsigned byte operation, CY is reset. If AH is not 0, CY is set.

When AH is a sign expansion of AL as a result of signed byte operation, CY is reset. In all other cases, CY is set.

If DW is 0 as a result of unsigned word operation, CY is reset. If DW is not 0, CY is set.

When DW is a sign expansion of AW as a result of signed word operation, CY is reset. In all other cases, CY is set.

In the case of 8-bit immediate operation, CY is reset if the product is within 16 bits. If the product is more than 16 bits, CY is set.

(4) Binary divide

Undefined

(5) Shift/rotate

In the case of shift/rotate including CY, CY is set if the bit to be shifted to CY is 1. If the bit is 0, CY is reset.

3.4.2 P (Parity Flag)

(1) Binary add/subtract, logical operation, and shift

P is set if there are an even number of bits in the lower 8 bits of the operation result. If there are an odd number of bits, P is reset.

When the result is all 0, P is set.

(2) Binary multiply/divide

Undefined

3.4.3 AC (Auxiliary Flag)

(1) Binary add/subtract

In the case of byte operation, AC is set if a carry is generated from the lower 4 bits to the higher 4 bits or a borrow is generated from the higher 4 bits to the lower 4 bits. In all other cases, AC is reset.

In word operation, the same operation is carried out about the lower bytes as in byte operation.

(2) Logical operation, binary operation, and shift/rotate

Undefined

3.4.4 Z (Zero Flag)

(1) Binary add/subtract, logical operation, and shift/rotate

In the case of byte operation, Z is set if all 8 bits of the byte operation result or all 16 bits of the word operation result are 0. In all other cases, Z is reset.

(2) Binary multiply/divide

Undefined

3.4.5 S (Sign Flag)

(1) Binary add/subtract, logical operation, and shift/rotate

In the case of byte operation, S is set if bit 7 of the result is 1. It is rest if bit 7 is 0.

In the case of word operation, S is set if bit 15 of the result is 1. It is reset if bit 15 is 0.

(2) Binary multiply/divide

Undefined

3.4.6 V (Overflow Flag)

(1) Binary add/subtract

In the case of byte operation, V is set if a carry from bit 7 differs from that from bit 6. If they are the same, V is reset.

In the case of word operation, V is set if a carry from bit 15 differs from that from bit 14. If they are the same, it is reset.

(2) Logical operation

V is reset regardless of the logical operation result.

(3) Binary multiply

If AH is 0 as a result of unsigned byte operation, V is reset. If AH is not 0, V is set.

When AH is a sign expansion of AL as a result of signed byte operation, V is reset. In all other cases, V is set.

If DW is 0 as a result of unsigned word operation, V is reset. If DW is not 0, V is set.

When DW is a sign expansion of AW as a result of signed word operation, V is reset. In all other cases, V is set.

In the case of 8-bit immediate operation, V is reset if the product is within 16 bits. If the product is more than 16 bits, V is set.

(4) Binary divide

V is reset regardless of the operation result.

(5) Shift/rotate

In the case of left 1-bit shift/rotate, the following occurs depending on the operation result.

When CY is the most significant bit, V is reset.
When CY is not the most significant bit, V is set.

In the case of right 1-bit shift/rotate, the following occurs depending on the operation result.

When the most significant bit is the lower bit following the most significant bit, V is reset.
When the most significant bit is not the lower bit following the most significant bit, V is set.

V becomes undefined in the case of multiple bit shift/rotate.

3.4.7 $\overline{\text{IBRK}}$ (I/O Break Flag)

Software interrupt generation with an input/output instruction in use is controlled.

If execution of an input/output instruction is attempted when $\overline{\text{IBRK}}$ = 0, a software interrupt (interrupt vector 6) is generated. This enables software to emulate the input/output instruction.

If an input/output is executed when $\overline{\text{IBRK}}$ = 1, no software interrupts are generated.

3.4.8 BRK (Break Flag)

If the BRK flag has been set, a software interrupt (interrupt vector 1) is generated when one instruction is executed. This enables trace to be executed for each instruction.

BRK is set with a memory manipulate instruction only if saved in the stack as part of PSW. If returned to PSW after it is set, BRK is validated.

3.4.9 IE (Interrupt Enable Flag)

IE is set with the EI instruction and enables interruption. It is reset with the DI instruction and disables interruption.

3.4.10 DIR (Direction Flag)

DIR is set with the SET1 DIR instruction and reset with the CLR1 DIR instruction.

If the DIR flag has been set, processing is carried out from the higher to lower addresses with a block transfer or input/output instruction. If DIR has been reset, processing is carried out from the lower to higher addresses.

3.4.11 RB0 to RB3 (Register Bank 0 to 3 Flag)

RB0 to RB3 indicate the register bank currently in use among 16 register banks set in the internal RAM area.

These flags are not returned from the stack with the POP PSW instruction.

Caution In the interrupt service routine, do not change the RB0 to RB3 values saved in the stack and PSW save register.

3.5 Memory Space

The V55PI has a 16-Mbyte memory space. The lower 1 Mbytes (000000H to 0FFFFFH) is accessible as basic memory space and 16 Mbytes (000000H to FFFFFFH) including the basic memory space is accessible as expansion memory space. The basic memory space is accessible using the segment register (PS, DS0, DS1, and SS) as is the case with V25 and V35. The expansion memory space is accessible with the expansion segment register (DS2 and DS3) and the basic memory space is mapped in the least significant 1 Mbytes. For details of the physical address, refer to **3.2.4 Segment registers (PS, SS, DS0, and DS1)** and **3.2.5 Expansion segment registers (DS2, DS3)**.

A group of registers with the on-chip peripheral hardware mode register and control register functions are mapped in the 496-byte space 0FFE00H to 0FFFEFH and are manipulated by memory access.

Independently of these spaces, there is a 512-byte register file space (internal RAM). The register file space is accessible with the register manipulate instruction as is the case with the V25 and V35.

Further, the register file space is also accessible as data memory by adding the dedicated prefix instruction (IRAM:) to the memory manipulate instruction.

If segment offset makes word reference at address FFFFH, the second reference address will be FFFFH plus one instead of 0000H in the same segment (address 0000H is referenced in the V20 and V30).

3.5.1 Basic memory space

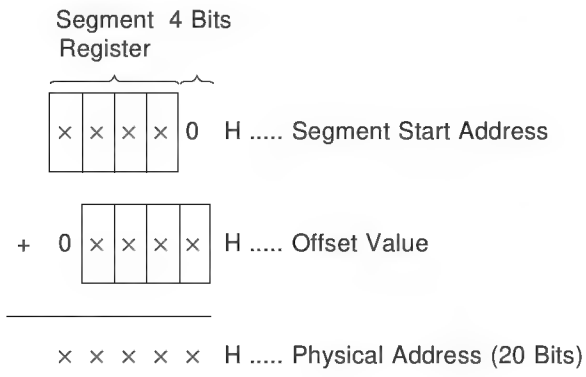
There are two memory spaces. They are a 1-Mbyte basic memory space and a 16-Mbyte expansion memory space. The basic memory space is mapped at the lower 1 Mbytes (000000H to 0FFFFFH) of the expansion memory space.

Figure 3-3 shows the 1-Mbyte basic memory space.

The conditions for accessing the basic memory space by software are the same as with V20, V30 and V25, V35.

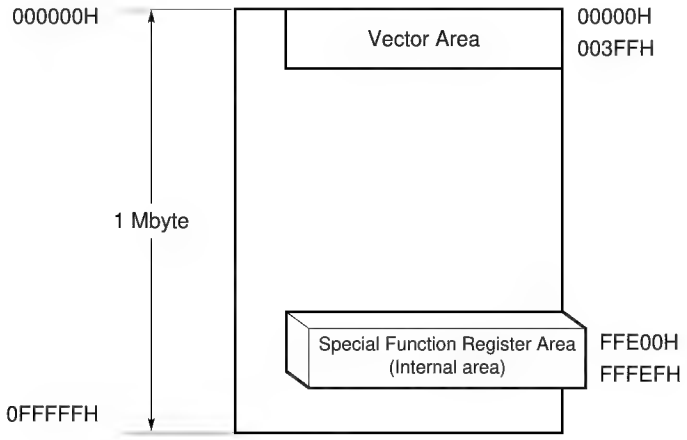
The physical address of the basic memory space is specified with the segment start address indicated by segment registers PS, SS, DS0, and DS1 and the offset value from the segment start position indicated by another register or immediate data.

The physical address of 1-Mbyte basic memory space is generated as follows.



The vector area for vectored interrupt and the special-function register area are mapped in the basic memory space. The external memory cannot make memory access to the external memory area overlapping the special-function register mapped area.

Figure 3-3. Basic Memory Space



The range 0FFFF0H to 0FFFFFH corresponds to the system boot program area. After a reset release, PS is set to 0FFFFH and PC is set to 0H, and therefore the program is executed from address 0FFFF0H.

3.5.2 Expansion memory space

Figure 3-4 shows the 16-Mbyte expansion memory space.

Only a data access is possible to the expansion memory space by software.

However, the basic memory space is mapped at the least significant 1 Mbytes (000000H to 0FFFFFH) and is accessible using segment registers PS, SS, DS0, and DS1.

In the expansion memory space, data access is possible using expansion segment registers DS2 and DS3. DS2 and DS3 can be specified as expansion segment override prefix by being added to the memory manipulate instruction.

The physical address of the expansion memory space is specified with the segment start address indicated by the expansion segment register and the offset value from the segment start position indicated by another register or immediate data. If the generated address indicates the least significant 1 Mbytes (000000H to 0FFFFFH), the basic memory space is accessed.

The physical address of 16-Mbyte expansion memory space is generated as follows.

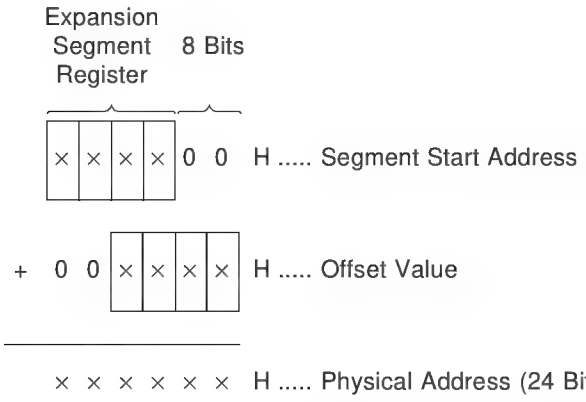
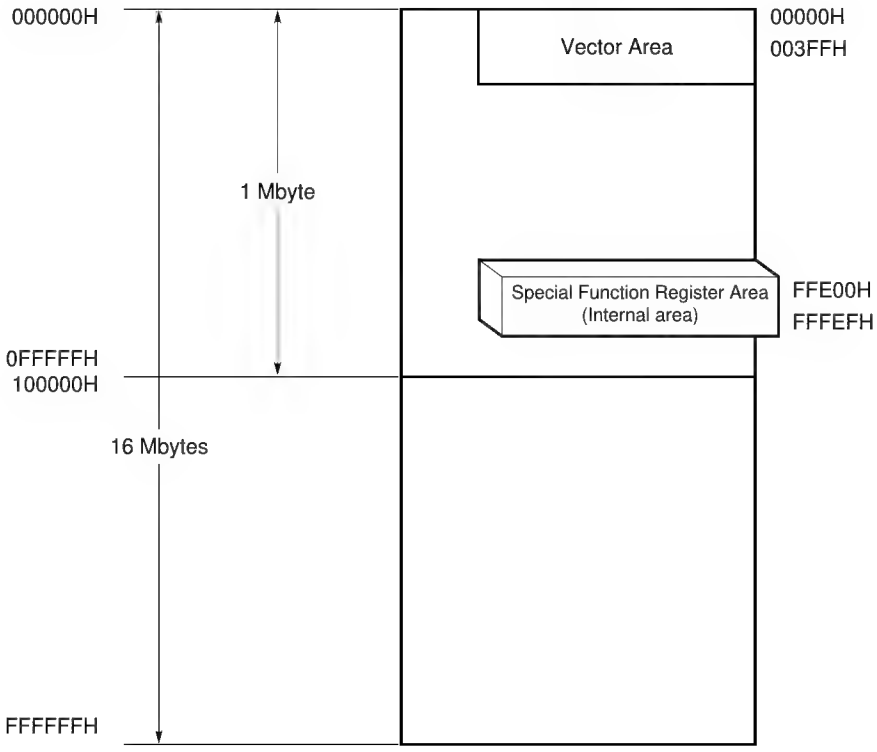


Figure 3-4. Expansion Memory Space



3.5.3 Special-function register area

A group of registers with the on-chip peripheral hardware operating mode specification and status monitoring functions are mapped in the 496-byte space 0FFE00H to 0FFFEFH.

No program fetch can be carried out from these areas.

The special-function register is manipulated by memory manipulation instruction access.

When the special-function register area is accessed, the address currently being accessed in the address bus is output. During write access to the special-function register, write data is output following address output. During read access, only the address is output, and the state then changes to high-impedance.

When the special-function register area is accessed, control signals such as $\overline{RD}$, $\overline{WRH}$, $\overline{WRL}$, $\overline{IORD}$, $\overline{IOWR}$ do not become active.

Table 3-5 gives a list of special-function registers. The items in the table have the following meanings.

Symbol ..... Special-function register name which corresponds to the operand coding format (symbol name) of the memory manipulation instruction.

R/W ..... Indicates whether the corresponding special-function register is enabled for read/write.
- R/W : Read/write enabled
- R : Read only enabled
- W : Write only enabled

Manipulating method ..... Indicates 1-bit, 8-bit, 16-bit, or 32-bit manipulation which can be carried out with each register.

After RESET ..... Indicates each register status after $\overline{RESET}$ input.

Caution Addresses not described correspond to the reserved area and should not be accessed by the user program.

Remark The lower 16 bits of the 21-bit (24-bit) special-function register are read/write enabled by word access only. The higher 5 bits (8 bits) are read/write enabled by either byte access or word access (the higher 8 bits become undefined in word access).

Table 3-5. List of Special-Function Registers (1/8)

Address	Special-Function Register Name	Symbol		R/W	Manipulatable Bit Unit				When Reset
					1 Bit	8 Bits	16 Bits	32 Bits	
0FFE00H	A/D conversion result register 0	ADCR0		R		√			Undefined
0FFE02H	A/D conversion result register 1	ADCR1		R		√			Undefined
0FFE04H	A/D conversion result register 2	ADCR2		R		√			Undefined
0FFE06H	A/D conversion result register 3	ADCR3		R		√			Undefined
0FFE10H	Parallel interface buffer	PAB		R/W[Note 1]		√			Undefined
0FFE18H	Parallel interface control register 0	PAC0		R/W	√	√			90H
0FFE19H	Parallel interface control register 1	PAC1		R/W	√	√			03H
0FFE1AH	Parallel interface status register	PAS		R/W[Note 2]		√			40H
0FFE1CH	Parallel interface acknowledge interval register 1	PAI1		W		√			Undefined
0FFE1DH	Parallel interface acknowledge interval register 2	PAI2		W		√			Undefined
0FFE20H	A/D converter mode register	ADM		R/W	√	√			00H
0FFEC0H	Interrupt mask flag register 0 (low)	MK0	MK0L	R/W	√	√	√		FFH
0FFEC1H	Interrupt mask flag register 0 (high)		MK0H	R/W	√	√			FFH
0FFEC2H	Interrupt mask flag register 1 (low)	MK1	MK1L	R/W	√	√	√		FFH
0FFEC3H	Interrupt mask flag register 1 (high)		MK1H	R/W	√	√			FFH
0FFEC4H	In-service priority register	ISPR		R	√	√			00H
0FFEC5H	Interrupt mode control register	IMC		R/W		√			80H
0FFEC9H	Interrupt request control register 09	IC09		R/W	√	√			43H
0FFECAH	Interrupt request control register 10	IC10		R/W	√	√			43H
0FFECBH	Interrupt request control register 11	IC11		R/W	√	√			43H

Notes 1. Depend on the input/output mode.

2. Some bits are R enabled and the other bits are R/W enabled.

Table 3-5. List of Special-Function Registers (2/8)

Address	Special-Function Register Name	Symbol	R/W	Manipulatable Bit Unit				When Reset
				1 Bit	8 Bits	16 Bits	32 Bits	
0FFECCH	Interrupt request control register 12	IC12	R/W	√	√			43H
0FFECDH	Interrupt request control register 13	IC13	R/W	√	√			43H
0FFECEH	Interrupt request control register 14	IC14	R/W	√	√			43H
0FFED0H	Interrupt request control register 16	IC16	R/W	√	√			43H
0FFED1H	Interrupt request control register 17	IC17	R/W	√	√			43H
0FFED2H	Interrupt request control register 18	IC18	R/W	√	√			43H
0FFED3H	Interrupt request control register 19	IC19	R/W	√	√			43H
0FFED4H	Interrupt request control register 20	IC20	R/W	√	√			43H
0FFED5H	Interrupt request control register 21	IC21	R/W	√	√			43H
0FFED6H	Interrupt request control register 22	IC22	R/W	√	√			43H
0FFED7H	Interrupt request control register 23	IC23	R/W	√	√			43H
0FFED8H	Interrupt request control register 24	IC24	R/W	√	√			43H
0FFED9H	Interrupt request control register 25	IC25	R/W	√	√			43H
0FFEDAH	Interrupt request control register 26	IC26	R/W	√	√			43H
0FFEDBH	Interrupt request control register 27	IC27	R/W	√	√			43H
0FFEDCH	Interrupt request control register 28	IC28	R/W	√	√			43H
0FFEDDH	Interrupt request control register 29	IC29	R/W	√	√			43H
0FFEDEH	Interrupt request control register 30	IC30	R/W	√	√			43H
0FFEDFH	Interrupt request control register 31	IC31	R/W	√	√			43H
0FFEE0H	Interrupt request control register 32	IC32	R/W	√	√			43H
0FFEE4H	Interrupt request control register 36	IC36	R/W	√	√			43H
0FFEE5H	Interrupt request control register 37	IC37	R/W	√	√			43H

Table 3-5. List of Special-Function Registers (3/8)

Address	Special-Function Register Name	Symbol	R/W	Manipulatable Bit Unit				When Reset
				1 Bit	8 Bits	16 Bits	32 Bits	
0FFF00H	Port 0	P0	R/W	√	√			Undefined
0FFF01H	Port 1	P1	R	√	√			Undefined
0FFF02H	Port 2	P2	R/W	√	√			Undefined
0FFF03H	Port 3	P3	R/W	√	√			Undefined
0FFF04H	Port 4	P4	R/W	√	√			Undefined
0FFF05H	Port 5	P5	R/W	√	√			Undefined
0FFF06H	Port 6	P6	R	√	√			Undefined
0FFF07H	Port 7	P7	R/W	√	√			Undefined
0FFF08H	Port 8	P8	R/W	√	√			Undefined
0FFF0CH	Port read control register	PRDC	R/W	√	√			00H
0FFF0EH	Real-time output port	RTP	R/W	√	√			Undefined
0FFF10H	Port 0 mode register	PM0	R/W	√	√			FFH
0FFF12H	Port 2 mode register	PM2	R/W	√	√			FFH
0FFF13H	Port 3 mode register	PM3	R/W	√	√			FFH
0FFF14H	Port 4 mode register	PM4	R/W	√	√			FFH
0FFF15H	Port 5 mode register	PM5	R/W	√	√			FFH
0FFF17H	Port 7 mode register	PM7	R/W	√	√			FFH
0FFF18H	Port 8 mode register	PM8	R/W	√	√			FFH
0FFF22H	Port 2 mode control register	PMC2	R/W	√	√			00H
0FFF23H	Port 3 mode control register	PMC3	R/W	√	√			00H
0FFF24H	Port 4 mode control register	PMC4	R/W	√	√			00H
0FFF25H	Port 5 mode control register	PMC5	R/W	√	√			00H

Table 3-5. List of Special-Function Registers (4/8)

Address	Special-Function Register Name	Symbol		R/W	Manipulatable Bit Unit				When Reset
					1 Bit	8 Bits	16 Bits	32 Bits	
0FFF27H	Port 7 mode control register	PMC7		R/W	√	√			00H
0FFF28H	Port 8 mode control register	PMC8		R/W	√	√			00H
0FFF2CH	Real-time output port control register	RTPC		R/W	√	√			40H
0FFF2DH	Real-time output port delay specify register	RTPD		R/W	√	√			Undefined
0FFF2EH	Port 7 buffer (low)	P7L		R/W	√	√			Undefined
0FFF2FH	Port 7 buffer (high)	P7H		R/W	√	√			Undefined
0FFF30H	Timer control register 0	TMC	TMC0	R/W	√	√	√		00H
0FFF31H	Timer control register 1		TMC1	R/W	√	√			00H
0FFF32H	Timer output control register 0	TOC	TOC0	R/W	√	√	√		00H
0FFF33H	Timer output control register 1		TOC1	R/W	√	√			00H
0FFF34H	External interrupt mode register 0	INTM	INTM0	R/W	√	√	√		00H
0FFF35H	External interrupt mode register 1		INTM1	R/W	√	√			00H
0FFF40H	Timer register 0	TM0		R/W			√		00H
0FFF42H	Timer register 1	TM1		R/W			√		00H
0FFF44H	Timer register 2	TM2		R/W			√		00H
0FFF46H	Timer register 3	TM3		R/W			√		00H
0FFF48H	Timer capture register 00	CT00		R/W			√		Undefined
0FFF4AH	Timer capture register 01	CT01		R/W			√		Undefined
0FFF4CH	Timer compare register 00	CM00		R/W			√		Undefined
0FFF4EH	Timer compare register 01	CM01		R/W			√		Undefined
0FFF50H	Timer capture register 10	CT10		R/W			√		Undefined
0FFF52H	Timer compare register 10	CM10		R/W			√		Undefined

Table 3-5. List of Special-Function Registers (5/8)

Address	Special-Function Register Name	Symbol	R/W	Manipulatable Bit Unit				When Reset
				1 Bit	8 Bits	16 Bits	32 Bits	
0FFF54H	Timer compare register 11	CM11	R/W			√		Undefined
0FFF58H	Timer compare register 20	CM20	R/W			√		Undefined
0FFF5AH	Timer compare register 21	CM21	R/W			√		Undefined
0FFF5CH	Timer compare register 22	CM22	R/W			√		Undefined
0FFF5EH	Timer compare register 23	CM23	R/W			√		Undefined
0FFF60H	Watchdog timer mode register	WDM	R/W**Note 1**	√	√			00H
0FFF64H	Timer compare register 30	CM30	R/W			√		Undefined
0FFF66H	Timer compare register 31	CM31	R/W			√		Undefined
0FFF6CH	PWM register	PWM	R/W	√	√			Undefined
0FFF6DH	PWM control register	PWMC	R/W	√	√			00H
0FFF70H	Transmit baud rate generator register 0	TxBRG0	R/W	√	√			Undefined
0FFF71H	Receive baud rate generator register 0	RxBRG0	R/W	√	√			Undefined
0FFF72H	Prescaler register 0	PRS0	R/W	√	√			00H
0FFF73H	UART mode register 0/clocked serial interface mode register 0	UARTM0/CSIM0	R/W	(√)	√			00H
0FFF74H	UART status register 0/SBI control register 0	UARTS0/SBIC0	**Note 2/Note 3**	(√)	√			20H/00H
0FFF75H	UART transmit buffer 0/clocked serial I/O shift register 0	TxB0/SIO0	W		√			Undefined
0FFF76H	Receive buffer 0	RxB0	R		√			Undefined

Notes 1. A WDM register write is possible only by the RSTWDT instruction (only in 8-bit units).

2. Some bits are R enabled and the other bits are R/W enabled.

3. R or W enabled in bit units.

Remark ( ): Depends on the mode.

Table 3-5. List of Special-Function Registers (6/8)

Address	Special-Function Register Name	Symbol		R/W	Manipulatable Bit Unit				When Reset
					1 Bit	8 Bits	16 Bits	32 Bits	
0FFF78H	Transmit baud rate generator register 1	TxBRG1		R/W	√	√			Undefined
0FFF79H	Receive baud rate generator register 1	RxBRG1		R/W	√	√			Undefined
0FFF7AH	Prescaler register 1	PRS1		R/W	√	√			00H
0FFF7BH	UART mode register 1/clocked serial interface mode register 1	UARTM1/CSIM1		R/W	(√)	√			00H
0FFF7CH	UART status register 1	UARTS1		Note	(√)	√			20H
0FFF7DH	UART transmit buffer 1/clocked serial I/O shift register 1	TxB1/SIO1		W		√			Undefined
0FFF7EH	Receive buffer 1	RxB1		R		√			Undefined
0FFF7FH	Protocol select register	ASP		R/W	√	√			00H
0FFF80H	Terminal counter 0 (low)	TC0	TC0L	R/W			√	√	Undefined
0FFF82H	Terminal counter 0 (high)		TC0H	R/W		√	√		Undefined
0FFF84H	Terminal counter modulo register 0 (low)	TCM0	TCM0L	R/W			√	√	Undefined
0FFF86H	Terminal counter modulo register 0 (high)		TCM0H	R/W		√	√		Undefined
0FFF88H	DMA up-down counter 0 (low)	UDC0	UDC0L	R/W			√	√	Undefined
0FFF8AH	DMA up-down counter 0 (high)		UDC0H	R/W		√	√		Undefined
0FFF8CH	DMA compare register 0 (low)	DCM0	DCM0L	R/W			√	√	Undefined
0FFF8EH	DMA compare register 0 (high)		DCM0H	R/W		√	√		Undefined
0FFF90H	DMA memory address register 0 (low)	MAR0	MAR0L	R/W			√	√	Undefined
0FFF92H	DMA memory address register 0 (high)		MAR0H	R/W		√	√		Undefined

Note Some bits are R enabled and the other bits are R/W enabled.

Remark ( ): Depends on the mode.

Table 3-5. List of Special-Function Registers (7/8)

Address	Special-Function Register Name	Symbol		R/W	Manipulatable Bit Unit				When Reset
					1 Bit	8 Bits	16 Bits	32 Bits	
0FFF94H	DMA read-write pointer 0 (low)	DPTC0	DPTC0L	R/W			√	√	Undefined
0FFF96H	DMA read-write pointer 0 (high)		DPTC0H	R/W		√	√		Undefined
0FFF9CH	DMA mode register 0	DMAM0		R/W	√	√			E0H
0FFF9DH	DMA control register 0	DMAC0		R/W	√	√			00H
0FFF9EH	DMA status register	DMAS		R/W	√Note	√			00H
0FFFA0H	Terminal counter 1 (low)	TC1	TC1L	R/W			√	√	Undefined
0FFFA2H	Terminal counter 1 (high)		TC1H	R/W		√	√		Undefined
0FFFA4H	Terminal counter modulo register 1 (low)	TCM1	TCM1L	R/W			√	√	Undefined
0FFFA6H	Terminal counter modulo register 1 (high)		TCM1H	R/W		√	√		Undefined
0FFFA8H	DMA up-down counter 1 (low)	UDC1	UDC1L	R/W			√	√	Undefined
0FFFAAH	DMA up-down counter 1 (high)		UDC1H	R/W		√	√		Undefined
0FFFACH	DMA compare register 1 (low)	DCM1	DCM1L	R/W			√	√	Undefined
0FFFAEH	DMA compare register 1 (high)		DCM1H	R/W		√	√		Undefined
0FFFB0H	DMA memory address register 1 (low)	MAR1	MAR1L	R/W			√	√	Undefined
0FFFB2H	DMA memory address register 1 (high)		MAR1H	R/W		√	√		Undefined
0FFFB4H	DMA read-write pointer 1 (low)	DPTC1	DPTC1L	R/W			√	√	Undefined
0FFFB6H	DMA read-write pointer 1 (high)		DPTC1H	R/W		√	√		Undefined
0FFFBCH	DMA mode register 1	DMAM1		R/W	√	√			E0H
0FFFBDH	DMA control register 1	DMAC1		R/W	√	√			00H
0FFFE0H	Software timer counter	STC		R			√		Undefined
0FFFE2H	Software timer counter compare register	STMC		R/W			√		FFFFH

Note Bit clear enabled.

Table 3-5. List of Special-Function Registers (8/8)

Address	Special-Function Register Name	Symbol	R/W	Manipulatable Bit Unit				When Reset
				1 Bit	8 Bits	16 Bits	32 Bits	
0FFFE8H	Programmable wait control register 0	PWC0	R/W	√	√			EAH
0FFFE9H	Programmable wait control register 1	PWC1	R/W	√	√			AAH
0FFFEAH	Memory block control register	MBC	R/W	√	√			FCH
0FFFECH	Refresh mode register	RFM	R/W	√	√			77H
0FFFEEH	Standby control register	STBC	R/W Note 1	√	√			Undefined Note 2
0FFFEFH	Processor control register	PRC	R/W	√	√			EEH

Notes 1. The SBF bit of the standby control register can be set (1) but cannot be cleared (0) (Writing is possible only when "1").

2. After power-on reset: 00H, in all other cases: Unchanged

3.5.4 Vector table area

The 1-Kbyte area 00000H to 003FFH in the memory space has the corresponding interrupt routine start addresses generated by interrupt requests and break instruction for 256 vectors (4 bytes per vector).

In the initialized status, vectors 0 to 47 are reserved for on-chip peripheral and software interrupts characteristic of the V55PI. The hardware interrupt vector address excluding NMI of vectors 8 to 47 can be changed with V0 and V1 bits of the interrupt mode control register (IMC).

The area which is not used for the vector table can be used for other purpose.

Vector 0 (00000H) : Divide error
Vector 1 (00004H) : Single step
Vector 2 (00008H) : NMI instruction
Vector 3 (0000CH) : BRK3 instruction
Vector 4 (00010H) : BRKV instruction
Vector 5 (00014H) : CHKIND instruction
Vector 6 (00018H) : Input/output instruction
Vector 7 (0001CH) : FPO instruction/exception trap

When V1 = V0 = 0

Vector 8 (00020H) : INTWDT
Vector 9 (00024H) : INTP0
Vector 10 (00028H) : INTP1
Vector 11 (0002CH) : INTP2
Vector 12 (00030H) : INTP3
Vector 13 (00034H) : INTP4
Vector 14 (00038H) : INTP5
Vector 15 (0003CH) : System reserved
Vector 16 (00040H) : INTCM00
Vector 17 (00044H) : INTCM01
Vector 18 (00048H) : INTCM10
Vector 19 (0004CH) : INTCM11
Vector 20 (00050H) : INTCM21
Vector 21 (00054H) : INTCM31
Vector 22 (00058H) : INTD0 DMA#0_MAIN
Vector 23 (0005CH) : INTD0S DMA#0_SUB
Vector 24 (00060H) : INTD1 DMA#1_MAIN
Vector 25 (00064H) : INTD1S DMA#1_SUB
Vector 26 (00068H) : INTSER0
Vector 27 (0006CH) : INTSER1

Vector 28 (00070H) : INTSR0/INTCSI0
Vector 29 (00074H) : INTSR1/INTCSI1
Vector 30 (00078H) : INTST0
Vector 31 (0007CH) : INTST1
Vector 32 (00080H) : INTSIT
Vector 33 (00084H) : System reserved
Vector 34 (00088H) : System reserved
Vector 35 (0008CH) : System reserved
Vector 36 (00090H) : INTPAI
Vector 37 (00094H) : INTAD
Vector 38 (00098H) : System reserved
Vector 39 (0009CH) : System reserved
Vector 40 (000A0H) : System reserved
Vector 41 (000A4H) : System reserved
Vector 42 (000A8H) : System reserved
Vector 43 (000ACH) : System reserved
Vector 44 (000B0H) : System reserved
Vector 45 (000B4H) : System reserved
Vector 46 (000B8H) : System reserved
Vector 47 (000BCH) : System reserved

When V1 = 0, V0 = 1
Vector 72 (00120H) : INTWDT
Vector 73 (00124H) : INTP0
to to to
Vector 110 (001B8H) : System reserved
Vector 111 (001BCH) : System reserved

When V1 = 1, V0 = 0
Vector 136 (00220H) : INTWDT
Vector 137 (00224H) : INTP0
to to to
Vector 174 (002B8H) : System reserved
Vector 175 (002BCH) : System reserved

When V1 = V0 = 1
Vector 200 (00320H) : INTWDT
Vector 201 (00324H) : INTP0
to to to
Vector 238 (003B8H) : System reserved
Vector 239 (003BCH) : System reserved

Remark The system reserved area is reserved for function expansion in future product development. These areas can also be accessed in the same way as accessing other vector tables.

3.6 Register File Space

Figure 3-5 shows the register file space.

The register file space has a size of 512 bytes and can set a register set with a maximum of 16 banks.

The register file space is separated from the memory space.

The register file space is accessible using the register manipulate instruction as is the case with the V25 and V35. Further, the register file space is accessible as data memory by adding the dedicated prefix instruction (IRAM:) to the memory manipulate instruction.

When the IRAM: prefix instruction is added to the memory manipulate instruction, the CPU makes a data access by taking the lower 9 bits of the offset value of the memory address as a register file address. In this case, physical address addition with the segment register is not executed and the external bus cycle is not started either (It accesses the external bus cycle independently (asynchronously)).

Example

```
Label 1:  MOV IRAM:  [0024H], AW ..... <1>
          MOV [0056H], BW        ..... <2>
```

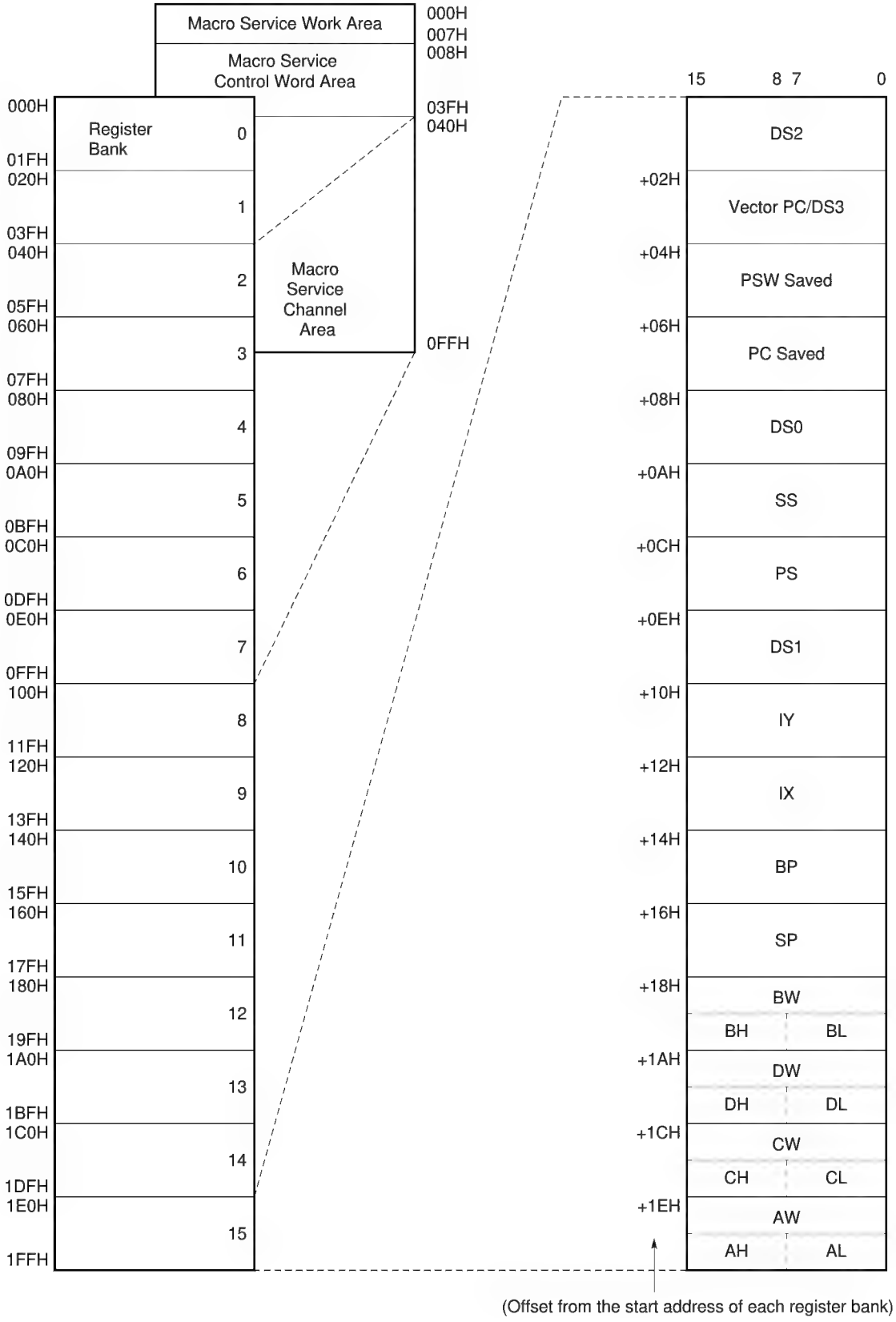
<1> Data is transferred to the register file space using the IRAM: prefix instruction. The AW register value is stored at address 24H of the register file.

<2> Instruction of data transfer to a memory space

When the IRAM prefix instruction is added to the primitive block transfer instruction and BCD operation instruction including the source block and destination block specification, the space is valid for the destination block.

The macro service control word area 008H to 03FH, macro service work area 000H to 007H and macro service channel use area 008H to 0FFH are assigned in the register file space in overlapped form. The work area can be used as any selected data area if a particular macro service (RTOPTRN) requiring work is not used.

Figure 3-5. Register File Space



3.7 I/O Space

The V55PI has a 64-Kbyte I/O space.

Figure 3-6 shows the I/O space map.

The I/O space is accessed using the address bus/data bus and control signal (such as $\overline{IORD}$, $\overline{IOWR}$).

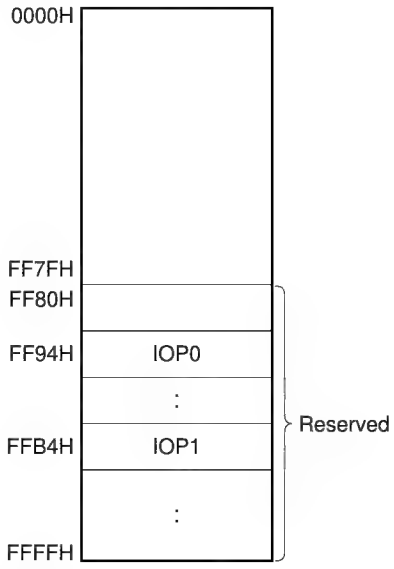
0 is output from the higher 8 bits of the unused address bus.

A wait cycle can be inserted in the I/O cycle by software and READY pin.

The I/O space FF80H to FFFFH is a reserved area where two input/output read-write pointers (IOP) of the peripheral DMA incorporated in the V55PI are allocated. The addresses of IOP0 and IOP1 are FF94H and FFB4H, respectively.

When the CPU executes an input/output instruction having the IOP address as operand, the DMA controller reads/writes data in its transfer buffer using the IOP contents as address values and automatically increments (or decrements) the IOP value in accordance with the DMA control registers contents. Thus, the data written with the DMA controller can be referred to with an input/output instruction and the data written with an input/output instruction can be transferred with the DMA controller.

Figure 3-6. I/O Map (64 Kbytes)



Remark IOPn corresponds to DMA read-write pointer n (DPTCn).

3.8 Logical Address and Physical Address

There are two types of addresses – logical address and physical address – in the memory space.

The physical address is an address that is directly allocated in the hardware. V55PI has a 16-Mbyte memory space. The low-order 1 Mbyte (000000H to 0FFFFFH) of this space is a basic memory space and a 16-Mbyte space (000000H to FFFFFFH) including the basic memory space is accessed as an expansion memory space. Therefore, the physical address ranges from 000000H to FFFFFFH. For details of the basic memory space and expansion memory space, refer to **3.5 Memory Space**.

A physical address is generated each time a bus control unit (BCU) which fetches instructions or transfers data is activated.

The logical address is an address that is used for address specification in the segment method.

3.8.1 Segment method

A segment is a small unit (max. 64 Kbytes) address space independent of programming. Each segment consists of consecutive memories and can be specified independently.

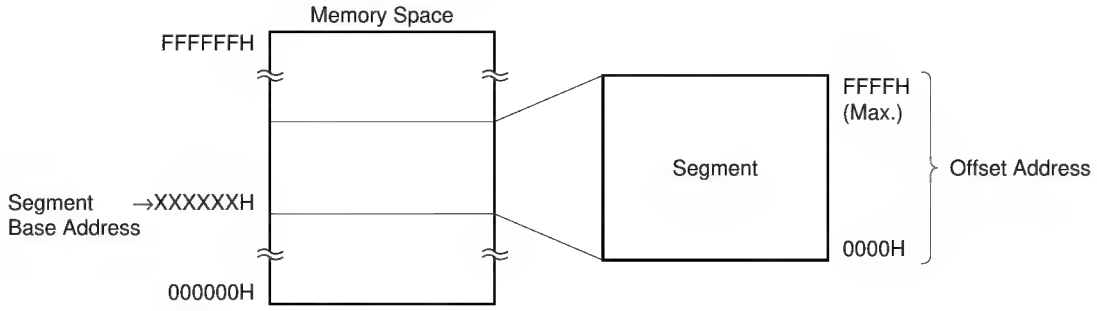
When executing programming using a machine language, the physical address can not be directly controlled. The memory address of V55PI is specified by the segment method.

The following two types of addresses are available for the segment method address specification.

Segment base address : Start address of segment (address based on the 16 Mbytes memory space)
Offset address : Address allocated for each segment

In the segment method, a segment base address is fixed as a reference. Only the offset address is used for processing in each segment.

Figure 3-7. Concept of Segment Method

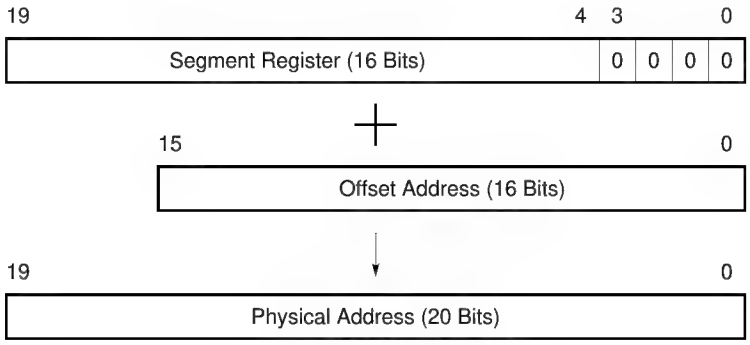


The segment base address is specified by the segment register.

The physical address is a sum of the segment base address and offset address.

The relationship among the segment register, offset address, and physical address is shown in Figure 3-8.

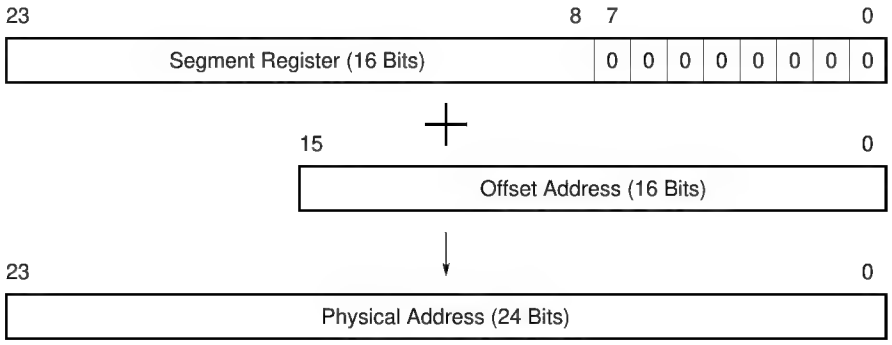
Figure 3-8. Relationship among the Segment Register, Offset Address, and Physical Address



As shown in Figure 3-8, the physical address (20 bits) is obtained by multiplying (16 times) (shifted to the left by 4 bits) the contents of the segment register (PS, SS, DS0, DS1) and by adding it to the offset value. In this case, the segment register contents and offset value is treated as an unsigned data.

For the expansion segment register (DS2, DS3), the physical address (24 bits) is obtained by multiplying (32 times) (shifted to the left by 8 bits) the contents of the expansion segment register and by adding it to the offset value.

Figure 3-9. Relationship among the Expansion Segment Register, Offset Address, and Physical Address



In a program consisting of multiple segments in which allocation addresses are specified by the logical address, each segment is independently compiled and assembled to form multiple object modules. Each object module has a segment name, size, content, control information, etc., and they serve as a parameter in link processing.

The multiple object modules are linked and a segment base address corresponding to the physical address is specified, and the multiple object modules are ready for loading to memory.

3.8.2 Segment configuration

Six segments (program, stack, data 0, data 1, data 2, and data 3) can be separately defined for the V55PI. Each segment is specified a start address using the following four segment registers and two expansion segment registers (Refer to **3.2.4 Segment registers (PS, SS, DS0, DS1)** and **3.2.5 Expansion segment registers (DS2, DS3)**.

BCU uses different segment registers for generating physical addresses according to the type of memory bus cycle.

Segment register

- Program segment register (PS)
- Stack segment register (SS)
- Data segment 0 register (DS0)
- Data segment 1 register (DS1)

Expansion segment register

- Expansion data segment 2 register (DS2)
- Expansion data segment 3 register (DS3)

The offset address in each segment is specified by a specific register or effective address. The relationship between the segment register and offset address specification is shown in Table 3-6.

Table 3-6. Relationship between Segment Register and Offset Address Specification

Segment register / Offset	Default	Override
PFP	PS	Disable
SP	SS	Disable
Effective address (BP base)		PS, DS0, DS1, DS2, DS3
Effective address (no BP base)	DS0	PS, SS, DS1, DS2
IX in Instruction set A[Note]		
IY in Instruction set B[Note]	DS1	DS3

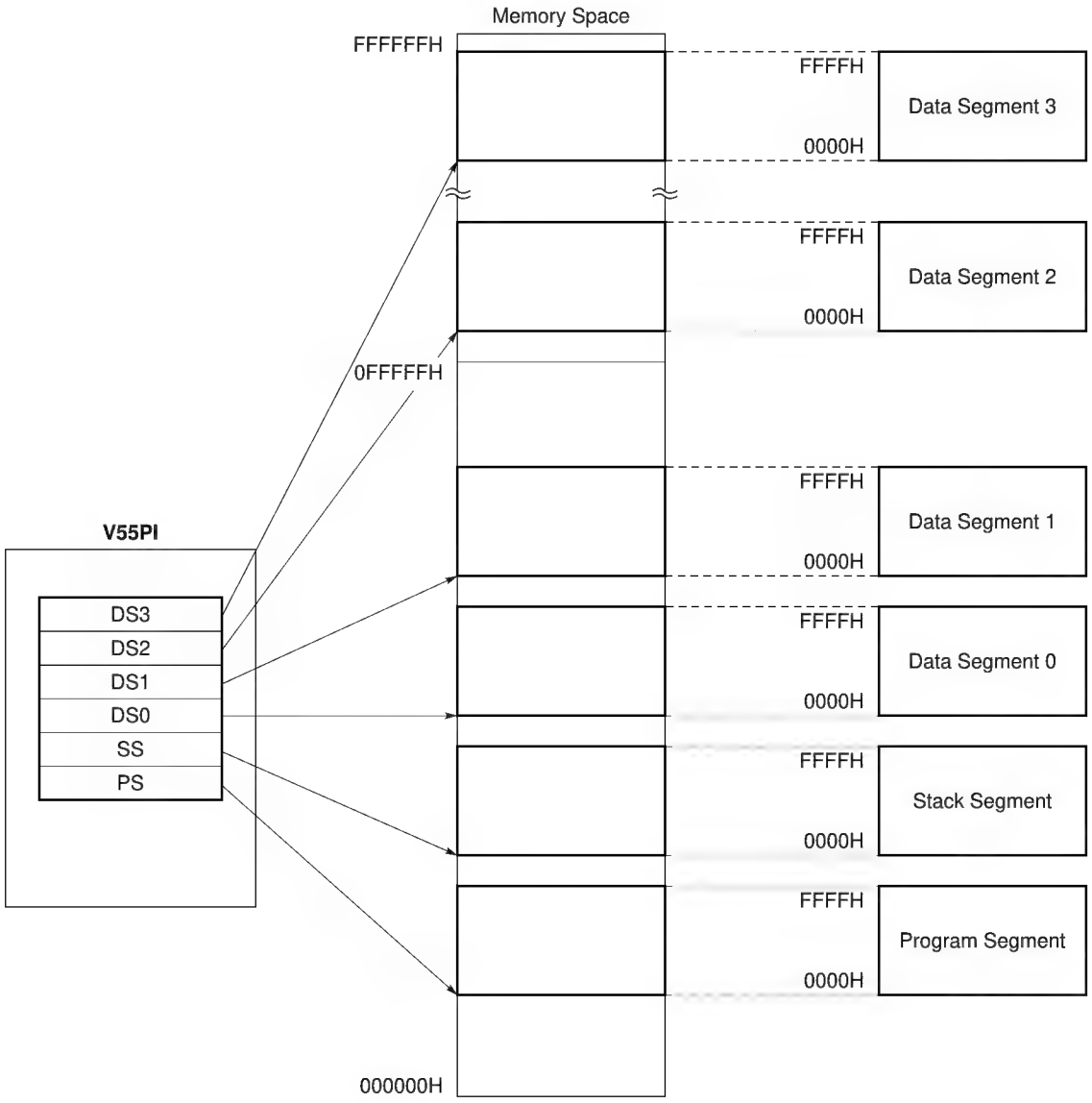
Note Instruction set A : Primitive block transfer instruction, primitive output instruction, BCD operation instruction, EXT instruction

Instruction set B : Primitive block transfer instruction, primitive input instruction, BCD operation instruction, INS instruction

If default offset is the prefetch pointer (PFP) and stack pointer (SP), the segment register which can be combined is fixed to PS and SS, and other segment registers can not be used.

In other default offsets, in addition to the default segment register, a specified segment register or expansion segment registers can be specified by the segment override prefix.

Figure 3-10. Relationship between Segment Register, Segment, and Memory Space



The contents of each segment is described below.

(1) Program segment

The start address of this segment is determined by the program segment register (PS), and the offset from the start address is specified by the prefetch pointer (PFP).

An instruction code, table data, etc. are allocated to this segment.

Using a segment override prefix (PS:), the program segment can be used as a generalized variable area and source data area during the instruction set A execution.

(2) Stack segment

The start address of this segment is determined by the stack segment register (SS), and the offset from the start address is specified by the effective address when the stack pointer (SP) and base pointer (BP) as a base address are used.

This segment is used as a stack area for the return address (contents of PS and PC), program status word (PSW), and general-purpose register contents, transmission/reception area, and local variable area while interrupt occurs or subroutine is processed.

Using a segment override prefix (SS:), the stack segment can be used as a generalized variable area and source data area during the instruction set A execution.

(3) Data segment 0

The start address of this segment is determined by the data segment 0 register (DS0), and the offset from the start address is specified by the effective address when BP is not used as a base address.

This segment is used as a storage area for generalized variables.

When the instruction set A is executed, this segment is used as a source data area. However, in this case, the contents of the index register (IX) become offset.

The stack segment is used as a default for the effective address when BP is used as a base address. Using a segment override prefix (DS0:), the data segment 0 can be used.

(4) Data segment 1

The start address of this segment is determined by the data segment 1 register (DS1). When the instruction set B is executed, this segment is used as a destination data area. However, in this case, the contents of the index register (IY) becomes offset.

Using a segment override prefix (DS1:), the data segment 1 can be used as a generalized variable area and source data area during the instruction set A execution.

(5) Data segment 2

The start address of this segment is determined by the expansion data segment 2 register (DS2), and the offset from the start address is specified by the effective address when BP is not used as a base address.
This segment is used as a storage area for generalized variables of the expansion memory space.
When the instruction set A is executed, this segment is used as a source data area. However, in this case, the contents of the index register (IX) becomes offset.
The stack segment is used as a default for the effective address when BP is used as a base address. Using an expansion segment override prefix (DS2:), the data segment 2 can be used.

(6) Data segment 3

The start address of this segment is determined by the expansion data segment 3 register (DS3).
When the instruction set B is executed, this segment is used as a destination data area. However, in this case, the contents of the index register (IY) becomes offset.
Using an expansion segment override prefix (DS3:), the data segment 3 can be used as a generalized variable area and source data area during the instruction set B execution.

3.8.3 Dynamic relocation

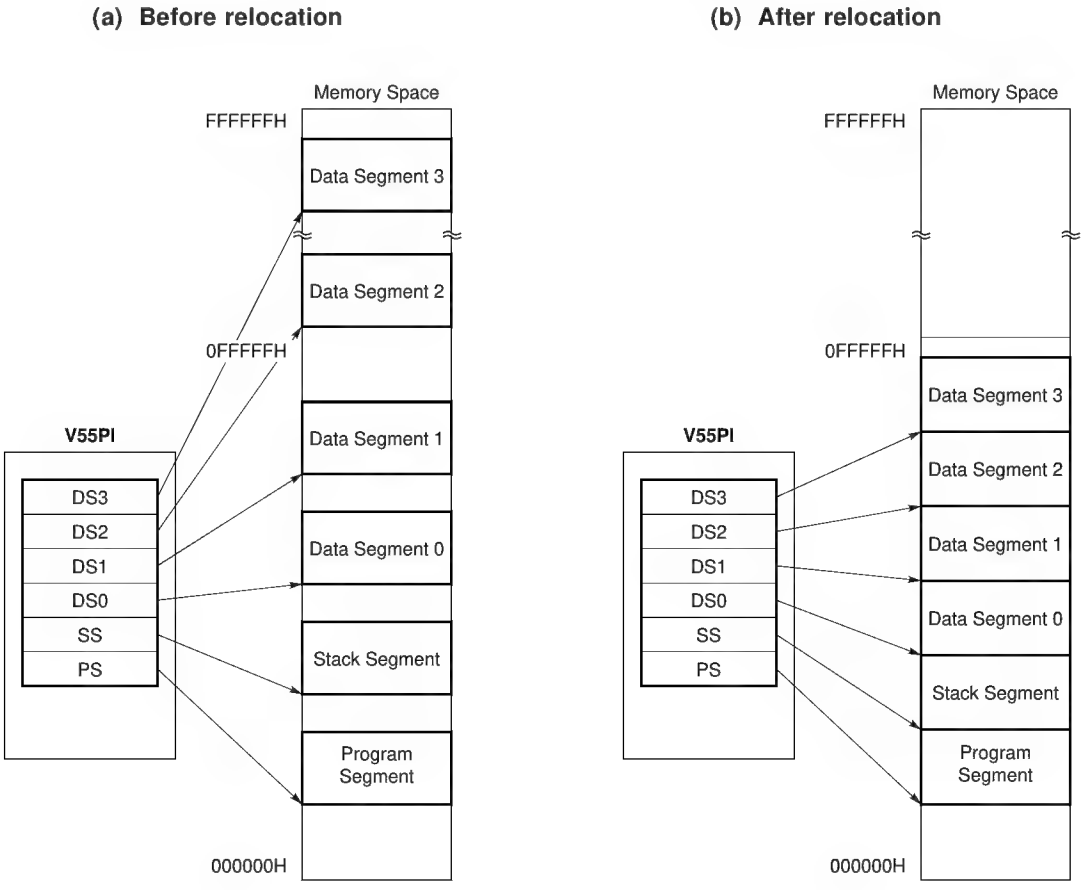
A dynamic relocation means that a program stored in several files is relocated to a vacant memory space each time the program is executed.

Figure 3-11 shows the concept of the dynamic relocation.

In the V55PI, the memory address of a program can be specified by an offset value for the each segment base address (specified by each segment register and expansion segment register). Therefore, simply by matching the contents of the segment register and expansion segment register of a program with the physical address of the memory to which the program is to be allocated, the program can be allocated to a desired memory space (However, this is possible only when the base address of each segment is not changed in the program.). As a result, programs can be allocated to the memory freely (address can be specified in 16-byte units), the memory is used more efficiently, and a system which executes several jobs and tasks can be easily created.

By using this, in order to execute programs in the files on external storage medium such as a floppy disk or hard disk, OS can manage a memory allocation area and type, segment register, and expansion segment register and load the programs in any vacant memory area for execution.

Figure 3-11. Dynamic Relocation



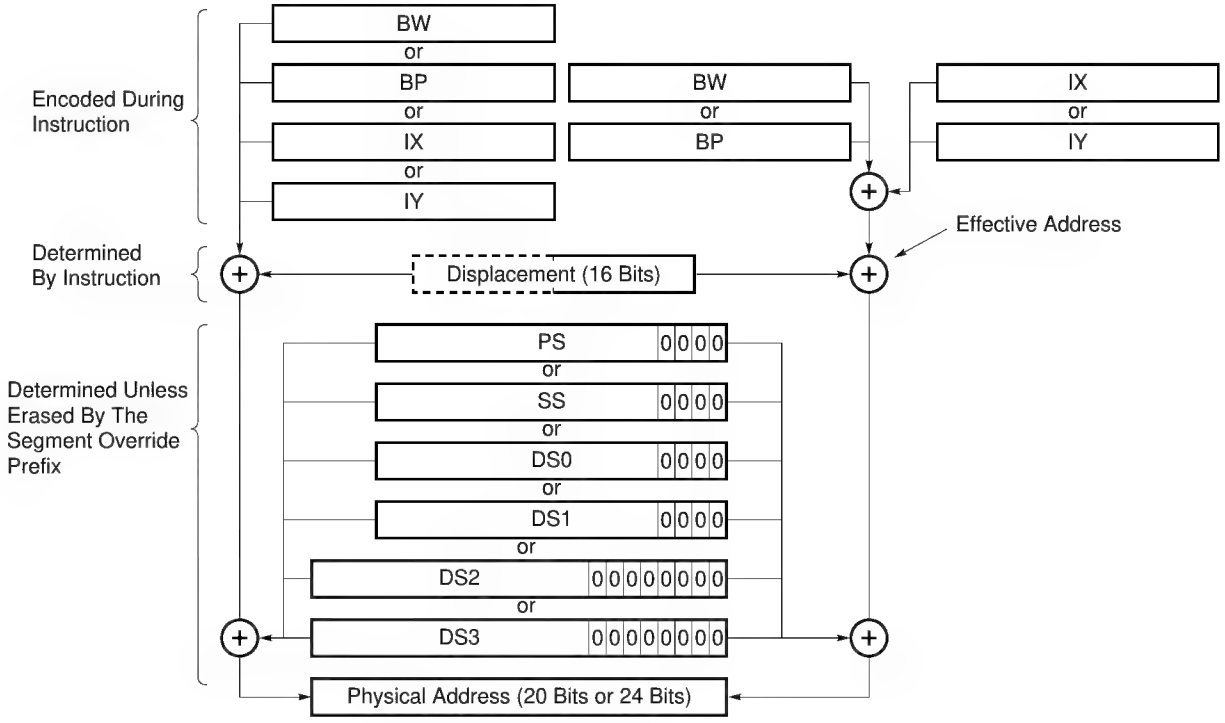
3.9 Effective Address

An effective address (EA) is an unsigned 16-bit number, which is a memory address to be instruction-processed represented in the offset value against the corresponding base address. When an instruction is executed, this address is operated by the execution unit (EXU) according to the instruction operand specification.

The EA is operated by the EXU in several different ways (addressing mode) and the way is specified by an instruction 2-byte operand. An information encoded to the instruction 2-byte operand indicates how the memory effective address expressed in an operand is operated by the EXU. This operand code is automatically generated from the program statement or instruction description using a compiler or assembler. All addressing mode can be used for the assembly language (Refer to **3.10 Addressing Mode**).

An operation way by EA is described below. In Figure 3-12, EXU operates EA by adding the contents of the displacement base register to that of the index register. These three elements can be combined freely for any instructions. The displacement is an 8-bit or 16-bit immediate number expressed in an operand.

Figure 3-12. Memory Address Operation



3.10 Addressing Mode

3.10.1 Instruction address

An instruction address is determined by the contents of PS and PC. Usually, this address automatically increments according to the number of instruction byte fetched each time an instruction is executed. When instructions with branches are executed, a branch destination address information is branched to PC or both PS and PC by the following addressings.

When BP is used for memory addressings other than the primitive instruction, the default segment register becomes DS0.

The default segment register of the destination block becomes DS1 during primitive instruction memory addressings. The default segment register of the source block for memory addressing becomes DS0.

(1) Direct addressing

A 2-byte or 4-byte immediate data in an instruction byte is directly loaded to PC, or PS and PC to become a branch address.

This is used for executing the following instructions.

```
CALL far-proc
CALL memptr16
CALL memptr32
BR   memptr16
BR   memptr32
```

(2) Relative addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to PC to form a branch address.

For 8-bit displacement, it is sign-extended to 16-bit data and it is added to PC.

When added, the contents of PC represents the next instruction start address.

This is used for executing the following instructions.

```
CALL near-proc
BR   near-label
BR   short-label
Conditional branch instruction   short-label
```

(3) Register addressing

The contents of any 16-bit register which is specified by the register specification field (3-bit) in the instruction byte is loaded as a branch address to PC.

Unlike data, all eight 16-bit registers (AW, BW, CW, DW, IX, IY, SP, BP) can be used.

This is used for executing the following instructions.

```
CALL regptr16
BR   regptr16
```

(4) Register indirect addressing

The contents of memory (word or double word) which is addressed by the 16-bit register (IX, IY, BW) specified by the register specification field in the instruction byte is loaded as a branch address to PC (or PC and PS).

Description examples

```
CALL memptr16          CALL WORD  PTR [IX]
CALL memptr32          CALL DWORD PTR [IY]
BR   memptr16          BR   WORD  PTR [BW]
BR   memptr32          BR   DWORD PTR [IX]
```

Remark When specified WORD PTR, a memptr16 instruction code is generated using the assembler. When specified DWORD PTR, a memptr32 instruction code is generated using the assembler.

(5) Indexed addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (IX and IY) serving as an index register. As a result, the contents of memory addressed (word or double word) are loaded as a branch address to PC.

This is used for executing the following instructions.

Description examples

```
CALL memptr16          CALL var [IX] [2]
CALL memptr32          CALL var [IY]
BR   memptr16          BR   var [BW]
BR   memptr32          BR   var [IX + 4]
```

Remark When the variable "var" has a word attribute, a memptr16 instruction code is generated using the assembler. When the variable "var" has a double word attribute, a memptr32 instruction code is generated using the assembler.

(6) Based addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (BP or BW) serving as a base register. As a result, the contents of memory addressed (word or double word) are loaded as a branch address to PC.

This is used for executing the following instructions.

Description examples	
CALL memptr16	CALL var [BP + 2]
CALL memptr32	CALL var [BP]
BR memptr16	BR var [BW] [2]
BR memptr32	BR var [BP]

Remark When the variable "var" has a word attribute, a memptr16 instruction code is generated using the assembler. When the variable "var" has a double word attribute, a memptr32 instruction code is generated using the assembler.

(7) Based indexed addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (BP or BW) serving as a base register and 16-bit register (IX or IY) serving as an indexed register. As a result, the contents of memory addressed (word or double word) are loaded as a branch address to PC.

This is used for executing the following instructions.

Description examples	
CALL memptr16	CALL var [BP] [IX]
CALL memptr32	CALL var [BP + 2] [IY]
BR memptr16	BR var [BW] [2] [IX]
BR memptr32	BR var [BP + 4] [IY]

Remark When the variable "var" has a word attribute, a memptr16 instruction code is generated using the assembler. When the variable "var" has a double word attribute, a memptr32 instruction code is generated using the assembler.

3.10.2 Memory operand address

The following methods are available for addressing registers or memories those are manipulated for executing instructions.

(1) Register addressing

This method addresses a register which manipulates the contents of the register specification field (reg = 3-bit field, sreg = 2-bit field) in an instruction byte.

In reg, it is combined with a 1-bit (W) which specifies word or byte in the instruction byte and specifies 8 types word registers (AW, BW, CW, DW, BP, SP, IX, IY) and 8 types byte registers (AL, AH, BL, BH, CL, CH, DL, DH).

In sreg, 4 types segment registers (PS, SS, DS0, DS1) are specified.

In xsreg, 2 types expansion segment registers (DS2, DS3) are specified.

There is a case where an instruction operation code specifies a specific register.

This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
reg, reg'	AW, BW, CW, DW, SP, BP, IX IY, AL, AH, BL, BH, CL, CH, DL, DH
reg16, reg16'	AW, BW, CW, DW, SP, BP, IX, IY
reg8, reg8'	AL, AH, BL, BH, CL, CH, DL, DH
sreg	PS, SS, DS0, DS1
xsreg	DS2, DS3
acc	AW, AL

Description examples

When MOV reg, reg:

```
MOV BP, SP
MOV AL, CL
```

(2) Immediate addressing

1-byte or 2-byte immediate data in an instruction byte is a manipulation target as it is.
This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
imm	8/16-bit immediate data
imm16	16-bit immediate data
imm8	8-bit immediate data
pop-value	16-bit immediate data

In imm, identification of 8-bit and 16-bit is determined by the assembler by judging an imm value described for the operand or an attribute of another operand simultaneously described, and a word/byte specification bit W is determined.

Description example

When MOV reg, imm:
MOV AL, 5; byte

When MUL reg16, reg16, imm16:
MUL AW, BW, 1000H

(3) Direct addressing

An immediate data in an instruction byte addresses the manipulation target memory.
This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
mem	16-bit variable specifying 8/16 bit memory data
dmem	16-bit variable specifying 8/16 bit memory data
imm4	4-bit variable representing a bit field data bit length

Description example

When MOV mem, imm:
MOV WORD_BAR, 2000H

When MOV acc, dmem:
MOV AL, BYTE_BAR

(4) Register indirect addressing

A 16-bit register (IX, IY, BW) specified by the memory specification field (mod, mem) in an instruction byte addresses the manipulation target memory.

This addressing is used when instructions with the following operand description character are executed.

Description character

mem	[IX], [IY], [BW]

Description example

When SUB mem, reg:

SUB [IX], AW

(5) Auto increment/auto decrement addressing

This is a kind of register indirect addressing, however, it addresses a manipulation target according to the contents of the default register, then automatically increment/decrement the register contents (+1/–1 for byte processing, +2/–2 for word processing).

Therefore, using this addressing, an address updating for the next byte/word operand processing is automatically executed.

Identification of increment and decrement is determined by a direction flag (DIR). When DIR = 0, the content is incremented. When DIR = 1, the content is decremented.

This addressing is all executed for the default register, and used when instructions with the following operand description character are executed.

Description character	Default register
dst-block	IY
src-block	IX

This addressing is combined with a counter (CW) which counts the repetition processing number of byte/word operand to be used to control the block data processing.

(6) Indexed addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (IX or IY) serving as an indexed register. As a result, the manipulation target memory operand is addressed.

This addressing is suitable for accessing an array type data. The displacement specifies the array starting address and where is the contents of the indexed register on the array is determined.

This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
mem	var [IX], var [IY]
mem16	var [IX], var [IY]
mem8	var [IX], var [IY]

Description examples

TEST	When mem, imm:
TEST	BYTE_VAR [IX], 7FH
TEST	BYTE_VAR [IX + 8], 7FH
TEST	WORD_VAR [IX] [8], 7FFFH

Remark When the variable "var" has a byte attribute, a byte operand is specified. When the variable "var" has a word attribute, a word operand is specified. The corresponding instruction code is generated by an assembler respectively.

(7) Based addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (BP or BW) serving as a base register. As a result, the manipulation target memory operand is addressed.

This addressing is suitable for accessing an structure type data placed at several points on memory. The base register specifies the starting address of each structure and the displacement specifies one element of each structure.

This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
mem	var [BP], var [BW]
mem16	var [BP], var [BW]
mem8	var [BP], var [BW]

Description example

SHL	When mem, 1:
SHL	BYTE_VAR [BP], 1
SHL	WORD_VAR [BP + 2], 1
SHL	BYTE_VAR [BP] [4], 1

Remark When the variable "var" has a byte attribute, a byte operand is specified. When the variable "var" has a word attribute, a word operand is specified. The corresponding instruction code is generated by an assembler respectively.

(8) Based indexed addressing

A 1-byte or 2-byte immediate data in an instruction byte becomes a displacement value with a sign, and it is added to a 16-bit register (BP or BW) serving as a base register and a 16-bit register (IX or IY) serving as an index register. As a result, the manipulation target memory operand is addressed.

This addressing is very suitable for accessing a structure type data including an array type as it specifies one data by changing both the contents of the base register and index register. In other words, the base register specifies the start address of each structure, the displacement value identifies the offset amount from it to the start address of the array data, and the index register specifies the array data position.

This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
mem	var [Base register] [Index register]
mem16	var [Base register] [Index register]
mem8	var [Base register] [Index register]

Description example

PUSH	When mem16:
PUSH	WORD_VAR [BP] [IX]
PUSH	WORD_VAR [BP + 2], [IX + 6]
PUSH	WORD_VAR [BP] [4] [IX] [8]

(9) Bit addressing

A 3/4-bit immediate data in an instruction byte or low-order 3/4-bit of the CL register specifies the manipulation target 8/16-bit register or one bit of the memory.

By an instruction using this addressing, a specific one bit of the register/memory can be tested (discrimination of 0 or 1)/set/cleared/inverted regardless of other bit contents. Therefore, a byte/word data is not required in manipulating one bit such as performing setting/resetting using AND or OR instruction.

This addressing is used when instructions with the following operand description character are executed.

Description character	Description method
imm4	Word operand bit number
imm3	Byte operand bit number
CL	CL

Description example

TEST1	When reg8, CL:	TEST1 AL, CL
NOT1	When reg8, imm3:	NOT1 CL, 5
CLR1	When mem16, CL:	CLR1 WORD_VAR [IX], CL
SET1	When mem16, imm4:	SET1 WORD_VAR [BP], 9

(10) Special function register addressing

A 1-byte immediate data in an instruction byte becomes an offset value (without sign) from the start of a special function register area and address the manipulation target special function register.

This addressing mode is suitable for BTCLR and BTCLRL instructions.

Description character	Description method
sfr	8-bit variable that specifies 8-bit special function register (Address: FFF00H to FFFEFH).
sfr1	8-bit variable that specifies 8-bit special function register (Address: FFE00H to FFEFFH).

Description example

BTCLR	DMAS, 3, 45

Remark The special function register is mapped in the memory space and ranges from FFE00H to FFFEFH.

[MEMO]

CHAPTER 4 BUS CONTROL FUNCTION

The V55PI performs interface to external memory and external I/O via an external bus.

4.1 External Bus Control Pins

See **2.1.2. Non-port functions** for details of the external bus control pins.

When using the function of a pin which also services as port pin, it is necessary to select the corresponding function with the port mode control register (PMCn).

4.2 External Data Bus Width Selection Function

This function specifies whether external bus data input/output is performed in 8-bit units or 16-bit units.

The data bus width selection function is specified by manipulation of the D8/$\overline{D16}$ pin. If fixed at the high level, the external data bus width is set to 8 bits and if fixed at the low level, it is set to 16 bits. After a reset release, the D8/$\overline{D16}$ pin level is monitored and the external data bus width is determined.

D8/$\overline{D16}$ Pin	External Data Bus Width
Low level	16 bits
High level	8 bits

Caution When the bus width is 8 bits, pins AD8 to AD15 are maintained as address output during one bus cycle (except in hold mode). Also, the $\overline{WRL}$ pin is set to memory write strobe signal output and the $\overline{WRH}$ pin is fixed at the high level.

4.3 Basic Bus Cycle

The V55PI bus cycle basically (when no wait cycle is inserted) operates in 3 clock cycles. Part of the address output signal and the data input/output signal are multiplexed and address output and data input/output are performed by time division.

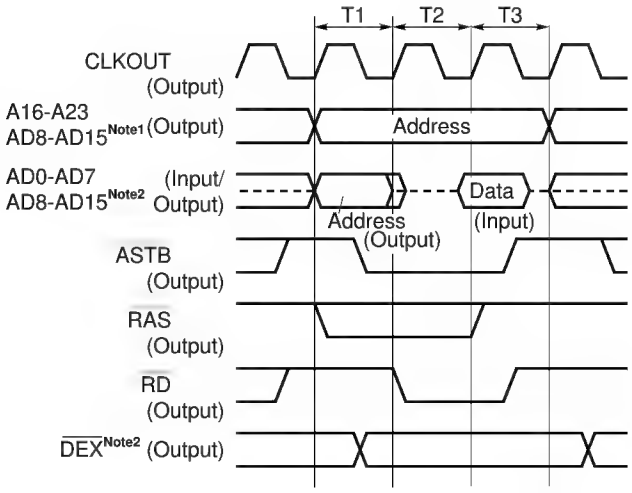
Address output is performed in the T1 cycle (first one clock cycle) which is an address cycle and data input/output is performed in the T2 and T3 cycles (remaining 2 clock cycles) which are data cycles.

The V55PI activates the following bus cycles for the external bus.

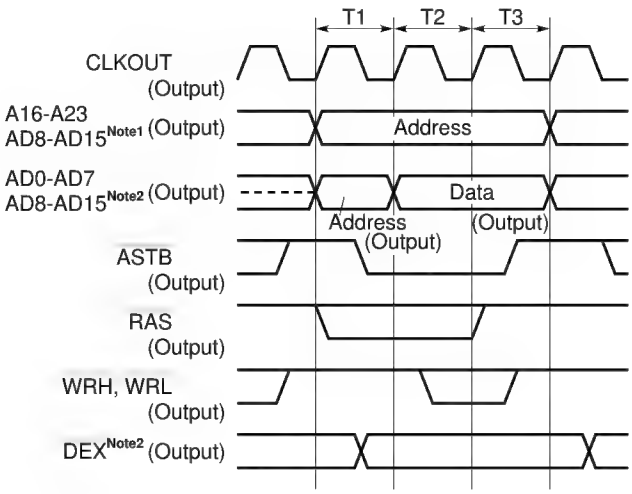
- Program code fetch cycle
- Memory read cycle
- Memory write cycle
- I/O read cycle
- I/O write cycle
- Refresh cycle
- DMA transfer cycle (external I/O to external memory)
- SFR read cycle
- SFR write cycle

The outline of timing chart for each cycle is shown in (1) to (6) below.
The state of the external bus when accessed using the special function register (SFR) is shown in (7) and (8).

(1) Memory read cycle, program code fetch cycle



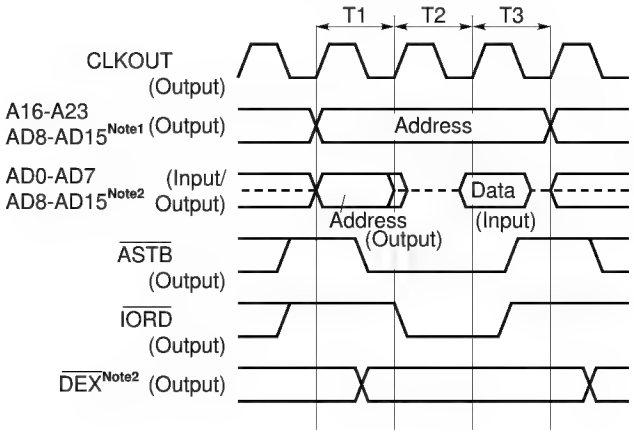
(2) Memory write cycle



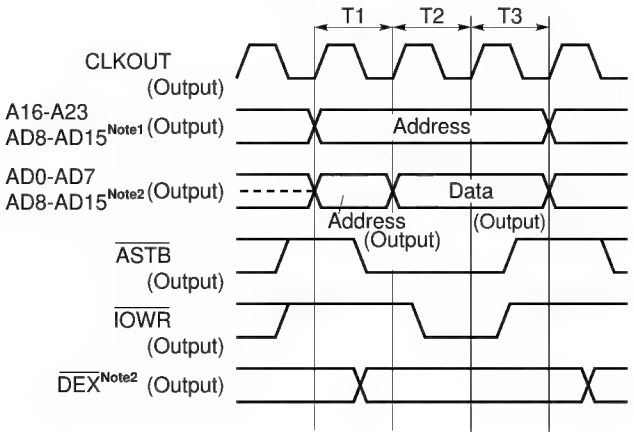
Notes 1. When external bus width 8 bits
2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

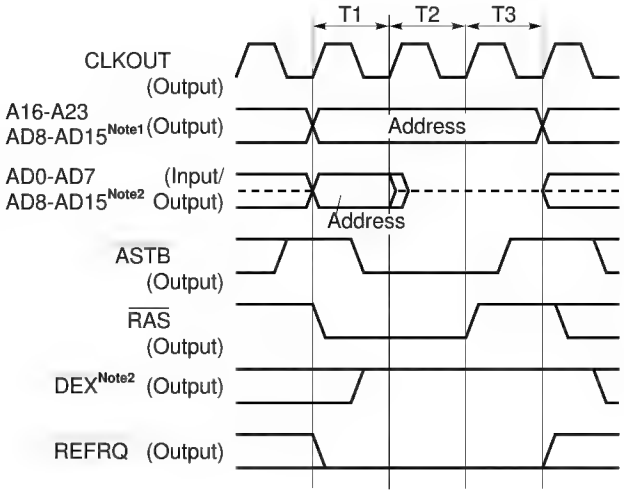
(3) I/O read cycle



(4) I/O write cycle



(5) Refresh cycle

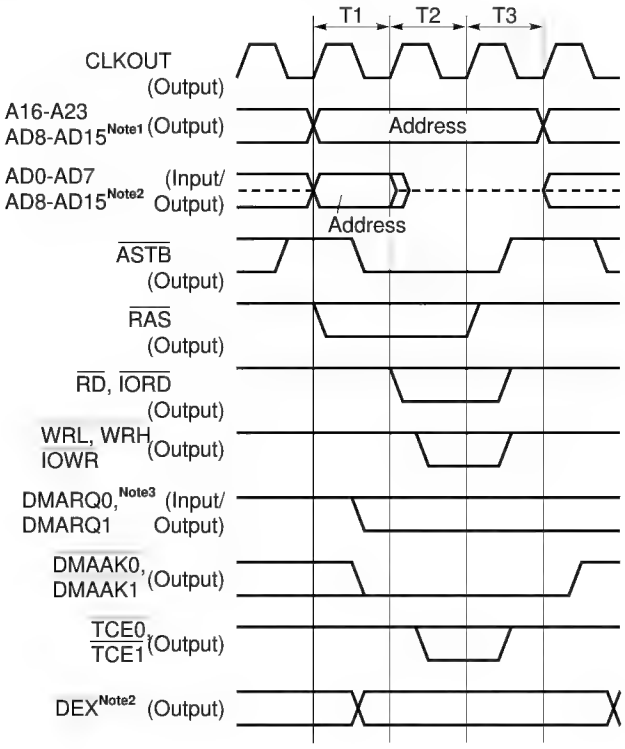


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

(6) DMA transfer cycle (external I/O to external memory)



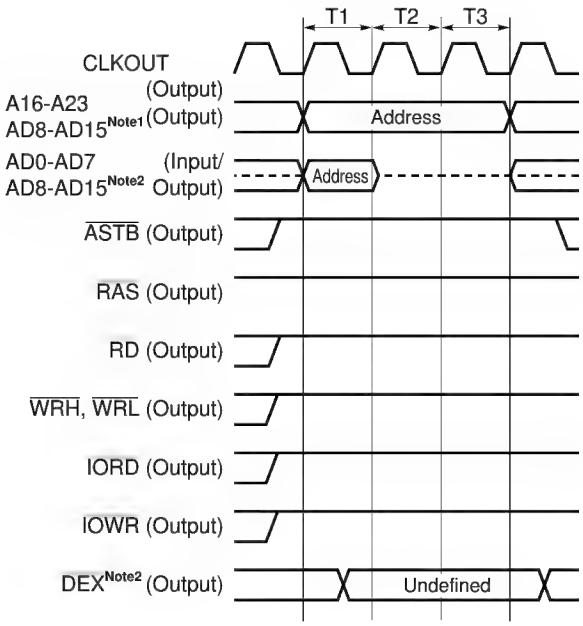
Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

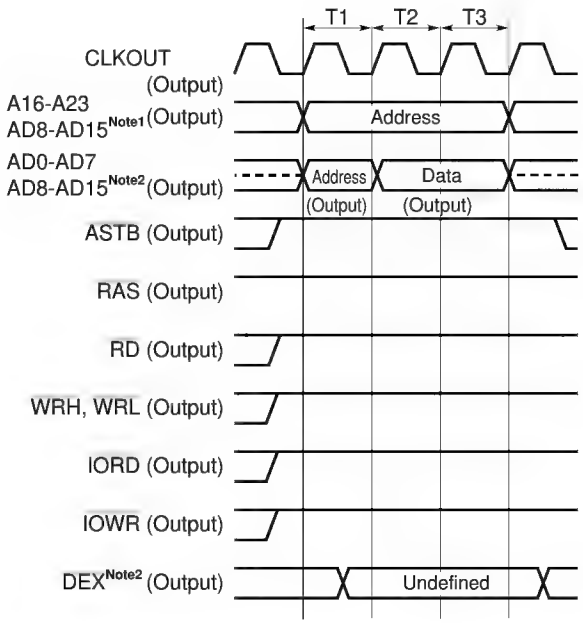
3. When demand release mode

Remark Dotted line indicates high-impedance.

(7) SFR read cycle



(8) SFR write cycle



Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

4.4 Memory Block Control Function

The V55PI has 1-Mbyte basic memory space and a 16-Mbyte expansion memory space. To connect an external memory corresponding to these memory spaces, the memory spaces can be divided into multiple memory blocks and controlled in the V55PI. For example, the basic memory space can be controlled after having been divided into memory blocks whose sizes can be specified piecemeal to cope with high-speed, small-capacity ROMs and SRAMs. The expansion memory space can be used as memory blocks to cope with low-speed, large-capacity ROMs and DRAMs.

4.4.1 Memory block control registers

In the V55PI the basic memory space 000000H to 0FFFFFH is divided into a maximum of 4 blocks, each variable in memory size, and the higher expansion memory space 100000H to FFFFFFH is divided into 2 blocks, each variable in memory size, and wait control is executed on each block. The memory size of each block of the basic memory space is specified with the memory block control register (MBC).

The MBC register is an 8-bit register to specify the division of the basic memory space into a maximum of 4 blocks (blocks 0 to 3) and the expansion memory space into 2 blocks (blocks 4 and 5).

The MBC register sets the upper limit values of blocks 0 to 2 and block 4 with 2 bits. The upper limit address of block 3 is fixed to 0FFFFFH.

If the specified value of the upper limit address of a block is smaller than that of the upper limit address of a lower block, the block range becomes 0 and the lower block is not generated.

	7	6	5	4	3	2	1	0
MBC	MB31	MB30	MB21	MB20	MB11	MB10	MB01	MB00

Upper Limit Address of Block 0

MB01	MB00	Upper Limit Address
0	0	000000H (0 K)
0	1	01FFFFH (128 K)
1	0	03FFFFH (256 K)
1	1	07FFFFH (512 K)

Upper Limit Address of Block 1

MB11	MB10	Upper Limit Address
0	0	03FFFFH (256 K)
0	1	07FFFFH (512 K)
1	0	0BFFFFH (768 K)
1	1	0FFFFFH (1 M)

Upper Limit Address of Block 2

MB21	MB20	Upper Limit Address
0	0	09FFFFH (640 K)
0	1	0BFFFFH (768 K)
1	0	0DFFFFH (896 K)
1	1	0FFFFFH (1 M)

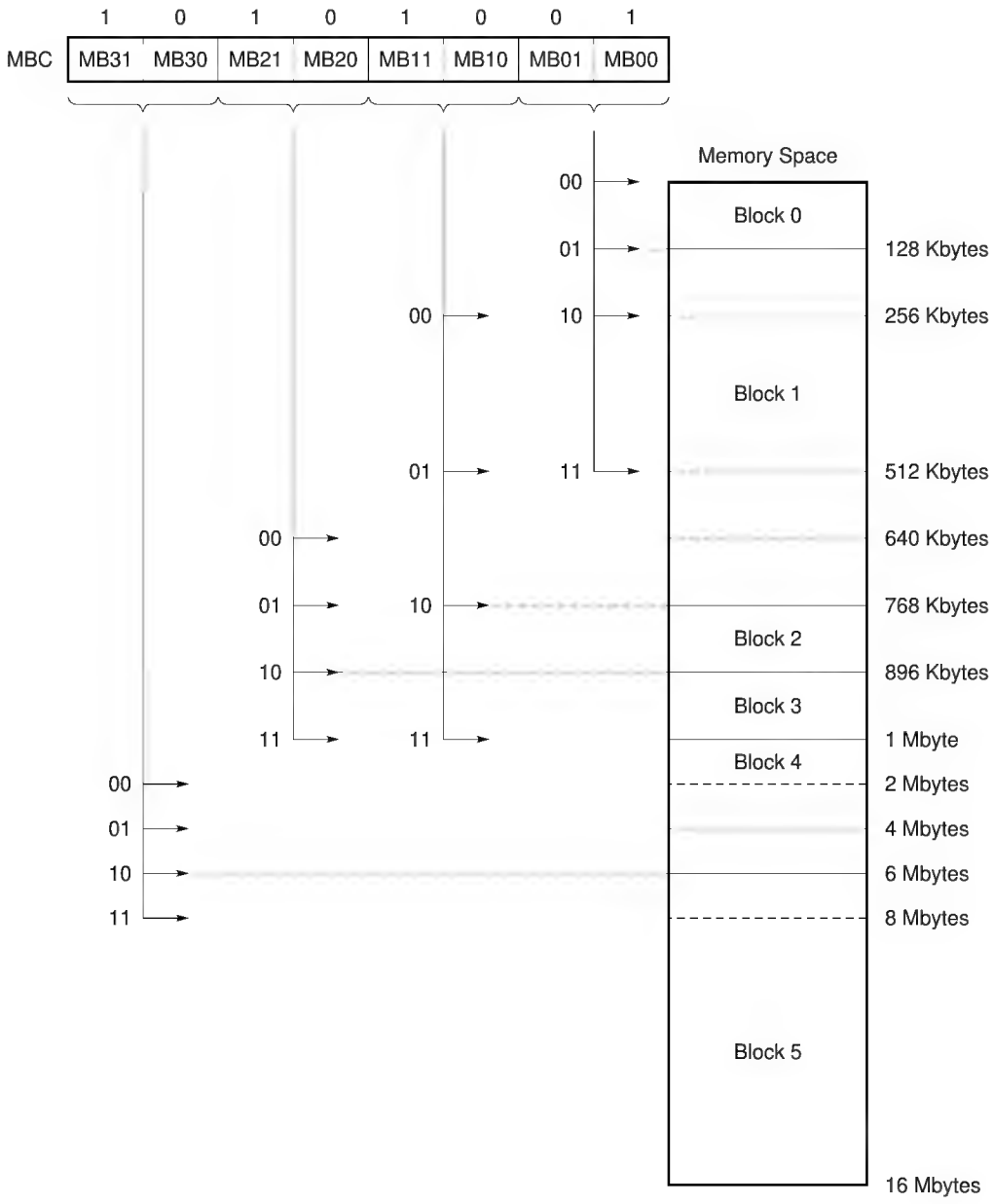
Upper Limit Address of Block 4

MB31	MB30	Upper Limit Address
0	0	1FFFFFH (2 M)
0	1	3FFFFFH (4 M)
1	0	5FFFFFH (6 M)
1	1	7FFFFFH (8 M)

Remark The shaded area indicates status after a reset.

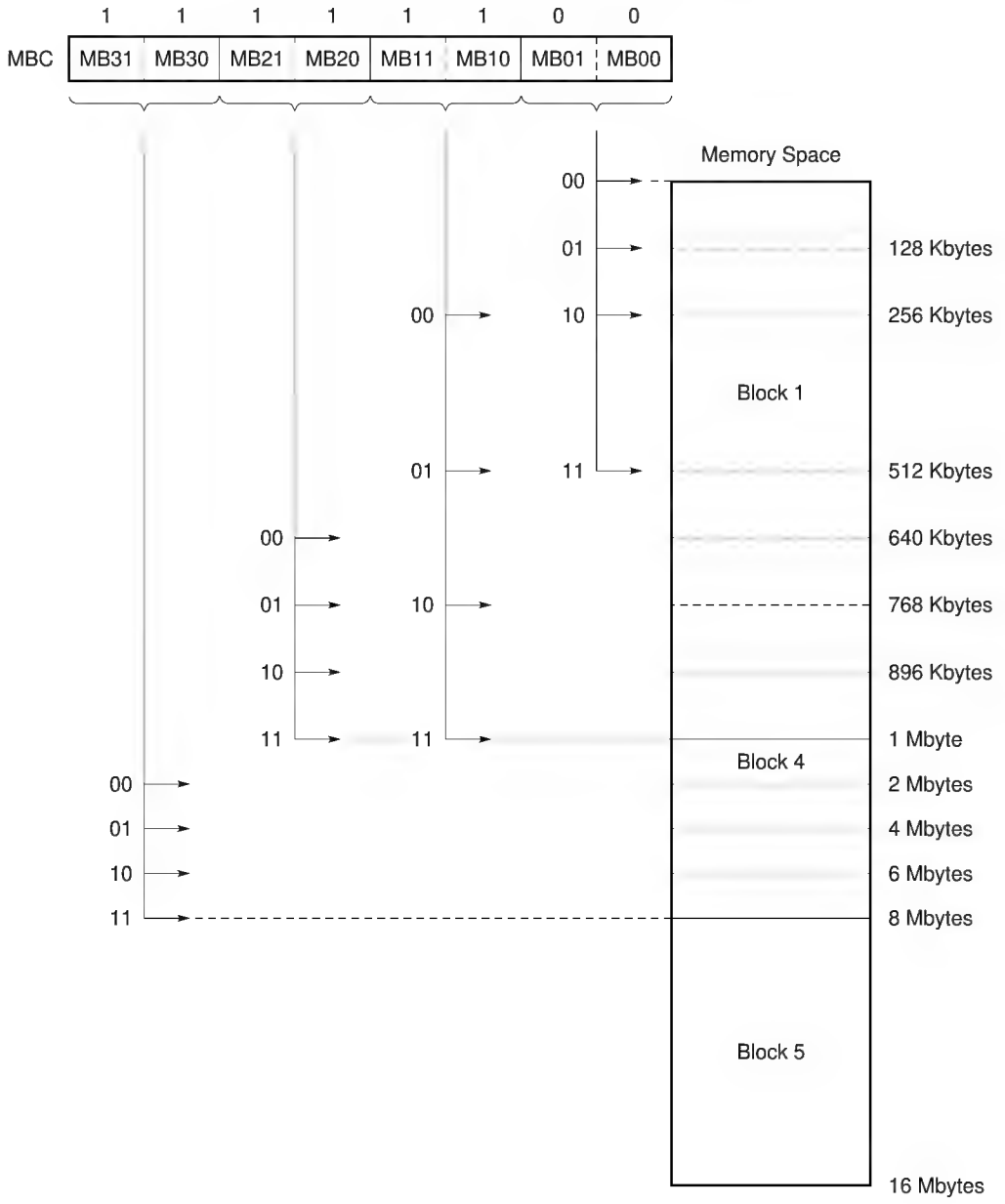
Figure 4-1 shows a memory block configuration when the MBC register value is set to A9H.

Figure 4-1. Memory Divided Control



The value of the MBC register is set to FCH after a reset. The memory block status after a reset is shown in Figure 4-2.

Figure 4-2. Memory Block Status after Reset



4.5 Wait Function

In the memory space, a wait state can be inserted to the external bus cycle in memory block units. A wait state can also be inserted in the I/O space.

Concerning the refresh cycle, wait cycles equivalent to the number of states specified for memory block 4 are inserted.

4.5.1 Address wait and data wait

There are two kinds of wait state, address wait and data wait, which can be inserted to the external bus cycle.

(1) Address wait

In the bus cycle, the address output period (T1 cycle) is extended by one clock cycle by inserting an address wait (see **Figure 4-10**). This is used to secure the DRAM precharge time.

(2) Data wait

This is inserted as a normal cycle between the T2 cycle and T3 cycle. The data input/output period in the bus cycle can be extended in 1-clock units (see **Figure 4-9**). This is used to secure the external memory and I/O access time.

4.5.2 Programmable wait function

In the V55PI, wait states can be inserted to the external bus cycle by software. However, address waits can be inserted only for memory block 1 and memory block 4. Specification of the data wait state can be performed by selecting from 0, 1, 2, 3 waits, and also additional control using the READY pin is possible. Specification of the number of wait states for each block is possible by the PWC0 register and PWC1 register.

(1) Programmable wait control registers (PWC0 and PWC1)

This register controls the number of waits states to be inserted in the bus cycle. An 8/1 bit access is possible for this register.

After a reset, PWC1 register value becomes AAH and PWC0 register value becomes EAH.

The shaded area indicates status after a reset.

	7	6	5	4	3	2	1	0	After Reset
PWC1	(BLOCK3)		(BLOCK2)		(BLOCK1)		(BLOCK0)		(AAH)
	DW31	DW30	DW21	DW20	DW11	DW10	DW01	DW00	

	7	6	5	4	3	2	1	0	After Reset
PWC0	BLOCK4	BLOCK1	(I/O space)		(BLOCK5)		(BLOCK4)		(EAH)
	AW1	AW0	IOW1	IOW0	DW51	DW50	DW41	DW40	

Data Wait (DW, IOW)

DWn1/IOW1	DWn0/IOW0	Wait State
0	0	0Note 1
0	1	1Note 2
1	0	2Note 2
1	1	3Note 2

Notes 1. The READY pin is ignored.

2. Additional control by the READY pin is also possible.

Address Wait (AW)

AWn		Wait State
AW0	0	Not inserted (block 1)
	1	Inserted (block 1)
AW1	0	Not inserted (block 4)
	1	Inserted (block 4)

4.5.3 Wait control with READY pin

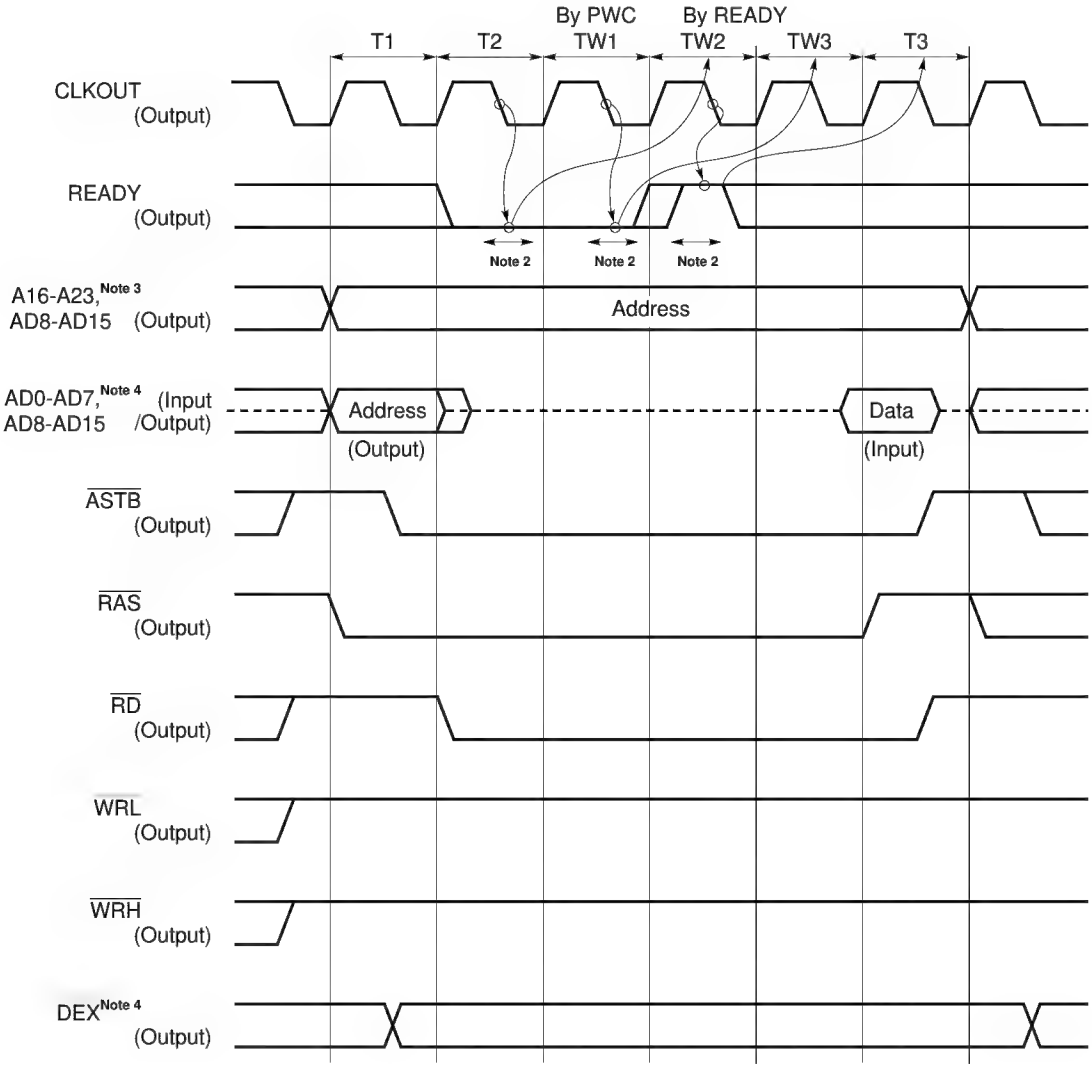
Wait state insertion with the READY pin is carried out independently of that set with the programmable wait control register (PWCn).

If the number of wait state insertions with the READY pin is larger than that specified for the PWCn register, wait state insertion with the READY pin is carried out. Conversely, if the former is smaller than the latter, wait state insertion specified for the PWCn register is carried out.

Caution Set the PWCn register to 1 or more wait state for wait state insertion with the READY pin. If the PWCn register is set to 0 wait, the READY signal is ignored and the wait state is not inserted.

Sampling of the READY pin is performed in synchronization with the CLKOUT fall timing in two bus cycles before the wait state inserted by the READY signal. Figures 4-3 and 4-4 show the sampling timings.

Figure 4-3. Memory Read Cycle Wait Insertion by READY Pin
(When there is no address wait and data wait by PWC register is 1[Note 1].)



Notes 1. When data wait is set to 2 by the PWC register, the READY signal in the T2 cycle is not sampled. The insertion of the TW3 cycle is determined according to only the READY state of the TW1 cycle and not the state of the T2 cycle.

In the same way, when data wait is set to 3 by the PWC register, the READY signal of the T2 cycle and TW1 cycle is not sampled.

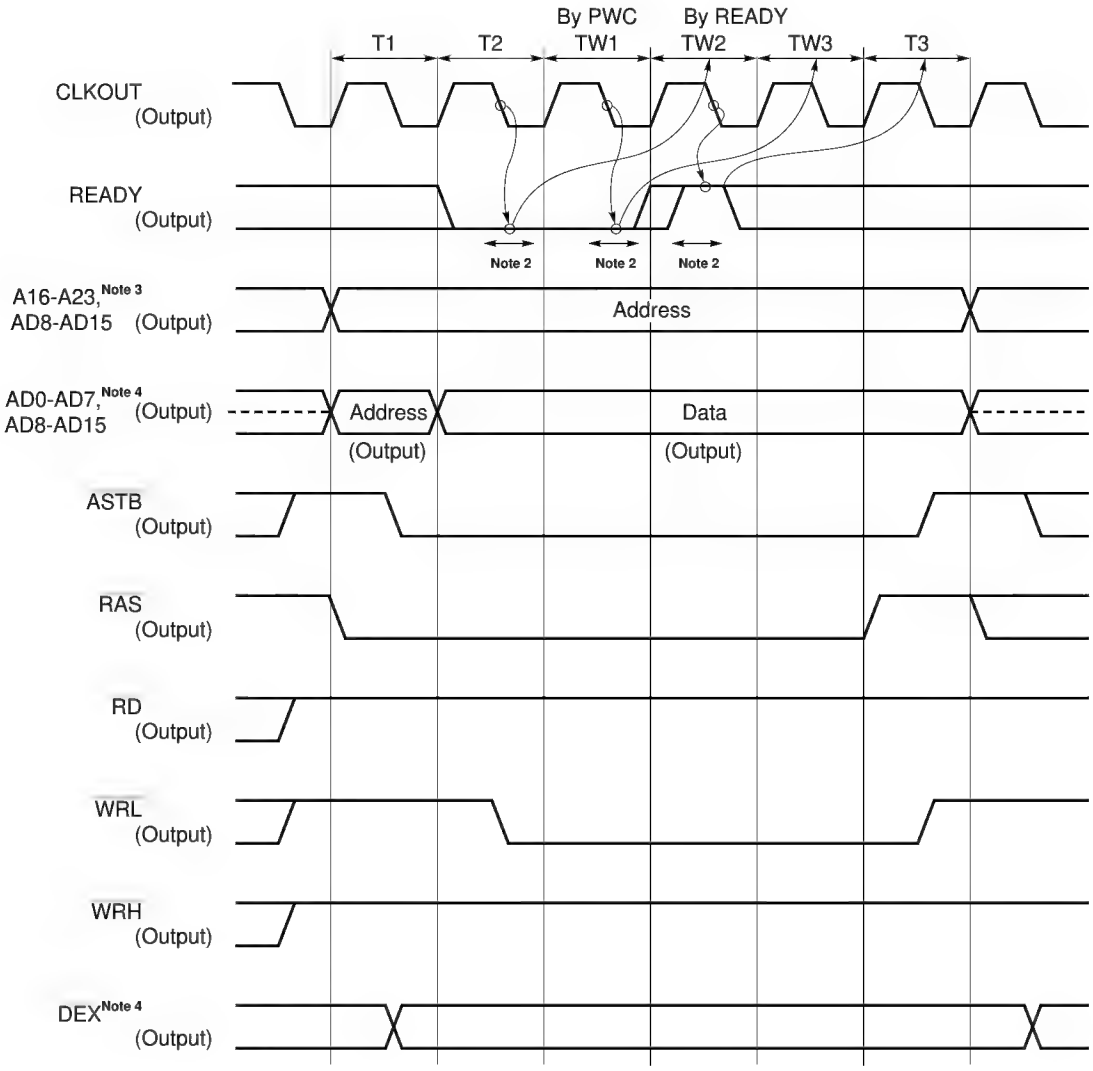
2. READY pin sampling

3. When external bus width 8 bits

4. When external bus width 16 bits

Remark Dotted line indicates high impedance.

Figure 4-4. Memory Write Cycle Wait Insertion by READY Pin (When there is no address wait and data wait by PWC register is 1[Note 1].)



Notes 1. When data wait is set to 2 by the PWC register, the READY signal in the T2 cycle is not sampled. The insertion of the TW3 cycle is determined according to only the READY state of the TW1 cycle and not the state of the T2 cycle.

In the same way, when data wait is set to 3 by the PWC register, the READY signal of the T2 cycle and TW1 cycle is not sampled.

2. READY pin sampling

3. When external bus width 8 bits

4. When external bus width 16 bits

Remark Dotted line indicates high impedance.

4.5.4 DMA transfer cycle wait

Wait control in DMA transfer of the external I/O-to-external memory follows the transfer destination or transfer source of address wait or data wait set with the greatest number of insertion cycles.

Example The number of waits in DMA transfer when the number of waits for memory and I/O are set as follows by the PWC register.

- Address wait (memory) : 1
- Data wait (memory) : 1
- Data wait (I/O) : 2

→

- Address wait: 1
- Data wait: 2

4.5.5 Wait in SFR access

When accessing SFR, one wait is normally inserted during both reading and writing. More than one wait is inserted in the following SFR registers.

Table 4-1. Wait in SFR Access (Above 1 Wait)

Function	Register Name	No. of Waits	
		Writing	Reading
Interrupt	Interrupt mask flag register (MK0, MK1)	1	2
	Interrupt request control register (IC09 to IC37)		
	Inservice priority register (ISPR)		
	Interrupt mode control register (IMC)		
	External interrupt mode register (INTM0, INTM1)		
Timer	Timer control register (TMC0, TMC1)	1, 2Note	2
	Timer output control register (TOC0, TOC1)	1	2
	Timer register (TM0 to TM3)	1 to 9Note	2 to 9Note
	Timer compare register (CM00, CM01, CM10, CM11, CM20, CM21, CM22, CM23, CM30, CM31) Timer capture register (CT00, CT01, CT10)	1, 2Note	2, 3Note
Serial Interface	Transmission baud rate generator register (TxBRG0, TxBRG1) Reception baud rate generator register (RxBRG0, RxBRG1)	1, 2Note	3, 4Note

Note The number of waits inserted differs according to the timing accessed.

4.6 $\overline{\text{RAS}}$ Pin Function

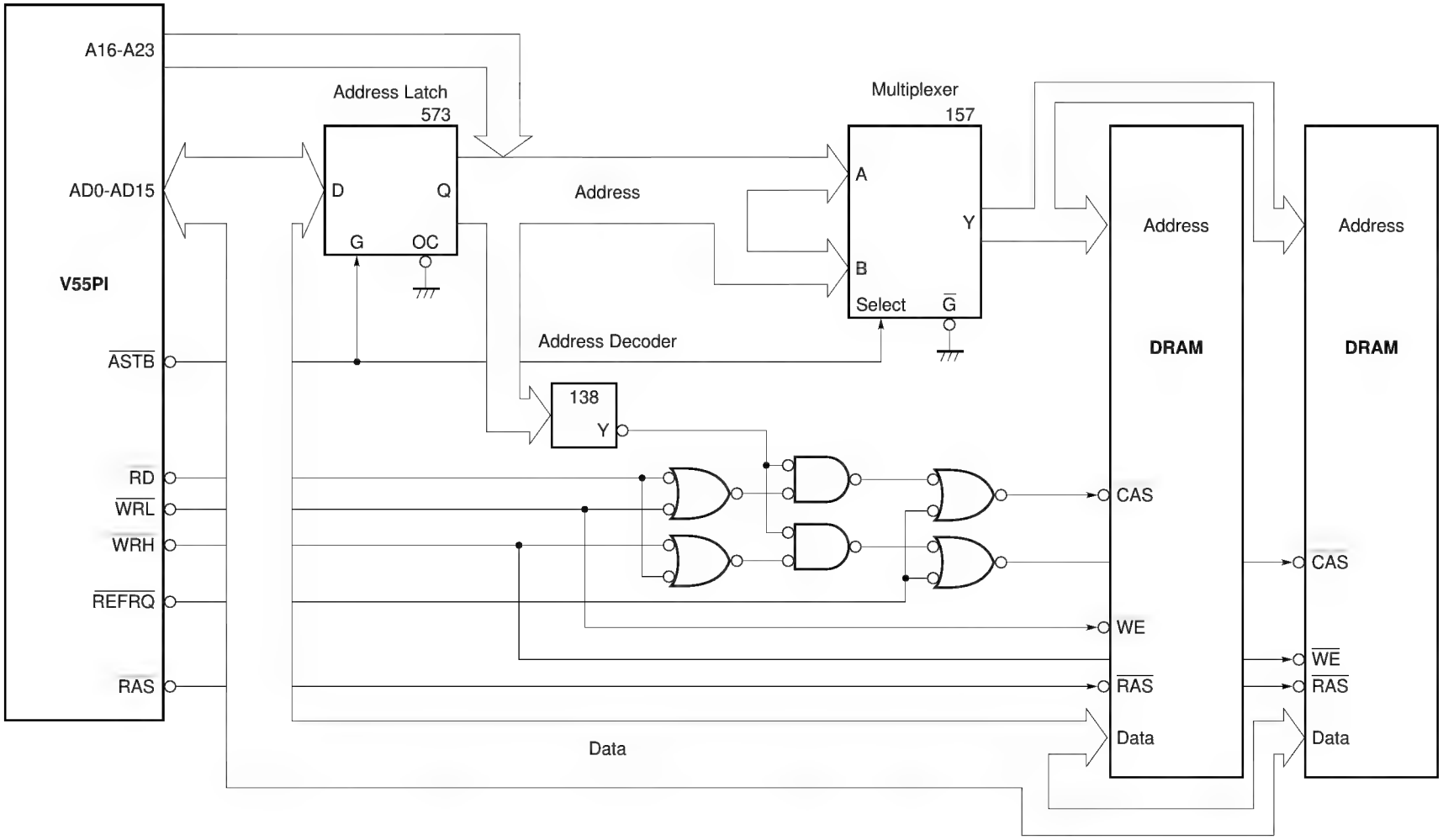
The $\overline{\text{RAS}}$ pin function is a function which gives the DRAM RAS timings to block 1 and block 4 using the $\overline{\text{RAS}}$ pin. The $\overline{\text{RD}}$, $\overline{\text{WRH}}$, and $\overline{\text{WRL}}$ pins give the DRAM CAS timing and the $\overline{\text{ASTB}}$ pin gives the DRAM row/column address switching timing.

The $\overline{\text{RAS}}$ pin is activated only in bus cycles (including refresh cycles) for block 1 and block 4 specified by the MBC register.

When connecting the $\overline{\text{RAS}}$ pin and using as the DRAM row address strobe signal, address wait must be inserted.

An example of DRAM connection circuit using the $\overline{\text{RAS}}$ pin is shown in Figure 4-5.

Figure 4-5. Example of DRAM Connection Circuit (Address Wait Insertion)



Remark The number of address/data bus signal lines, address latches, address decoders, and multiplexers depends on the size of the external memory.

4.7 Refresh Function

The V55PI has the following functions to refresh the DRAM, pseudo SRAM.

- Function of periodically inserting a refresh cycle in a series of bus cycles
- Refresh address output function to refresh the DRAM and pseudo SRAM
- Function of generating a refresh cycle in the HOLD or HALT mode
- Function of inserting a wait state in the refresh cycle

4.7.1 Refresh mode register (RFM)

The RFM register is an 8-bit register to control the refresh operation.

The refresh cycle can be selected using the output tap of the time base counter.

If, when a refresh request is generated while the previous refresh request is held pending because of other bus cycles, only the last refresh request is valid.

After a reset, the RFM register value becomes 77H.

	7	6	5	4	3	2	1	0
RFM	0	HLDRF	HLTRF	RFEN	0	RFT2	RFT1	RFT0

Refresh Cycle Specification

RFT2	RFT1	RFT0	Refresh Cycle	ϕ = 12.5 MHz	ϕ = 16 MHz
0	0	0	64/ϕ	5.12 μs	4.0 μs
0	0	1	128/ϕ	10.24 μs	8.0 μs
0	1	0	192/ϕ	15.36 μs	12.0 μs
0	1	1	256/ϕ	20.48 μs	16.0 μs
1	0	0	8/ϕ	0.64 μs	0.5 μs
1	0	1	16/ϕ	1.28 μs	1.0 μs
1	1	0	24/ϕ	1.92 μs	1.5 μs
1	1	1	32/ϕ	2.56 μs	2.0 μs

(ϕ: System clock)

Specification of Automatic Insertion of Refresh Cycle

RFEN	Automatic Insertion of Refresh Cycle
0	Automatic insertion disabled
1	Automatic insertion enabled

Specification of Automatic Insertion of Refresh Cycle in the HALT Mode

HLTRF	Automatic Insertion of Refresh Cycle
0	Automatic insertion disabled
1	Automatic insertion enabled Note 1

Specification of Automatic Insertion of Refresh Cycle in the HOLD Mode

HLDRF	Automatic Insertion of Refresh Cycle
0	Automatic insertion disabled
1	Automatic insertion enabled Notes 1, 2

Notes 1. When RFEN bit = 0, automatic insertion is disabled irrespective of the HLTRF and HLDRF bit contents.

2. When enabled, $\overline{\text{HLDAK}}$ output is forcibly started at high level at every refresh timing and a refresh cycle is inserted automatically.

Remark The shaded area indicates status after a reset.

4.7.2 Refresh cycle wait control

A wait state can be inserted in the refresh cycle.

The specified number of wait states and the number of wait states by READY pin are inserted for memory block 4 with the programmable wait control register (PWC0).

Refer to **4.5 Wait Function** for details of specification.

4.7.3 Refresh address

Buses AD0 to AD15 and A16 to A19 are activated in the refresh cycle.

The count-up is started every refresh cycle, by one for the external 8-bit bus width from X00000H to XFFFFFH, and by two for the external 16-bit bus width from ×00001H to ×FFFFFH.

After initialization by reset, the count-up is started from ×00000H for the external 8-bit bus width and from ×00001H for the external 16-bit bus width.

In the case of the external 16-bit bus width, the refresh address LSB (A0) is fixed at 1 and the $\overline{DEX}$ pin output is also fixed at 1.

A20 to A23 remain undefined during the refresh cycle.

4.7.4 Refresh timing

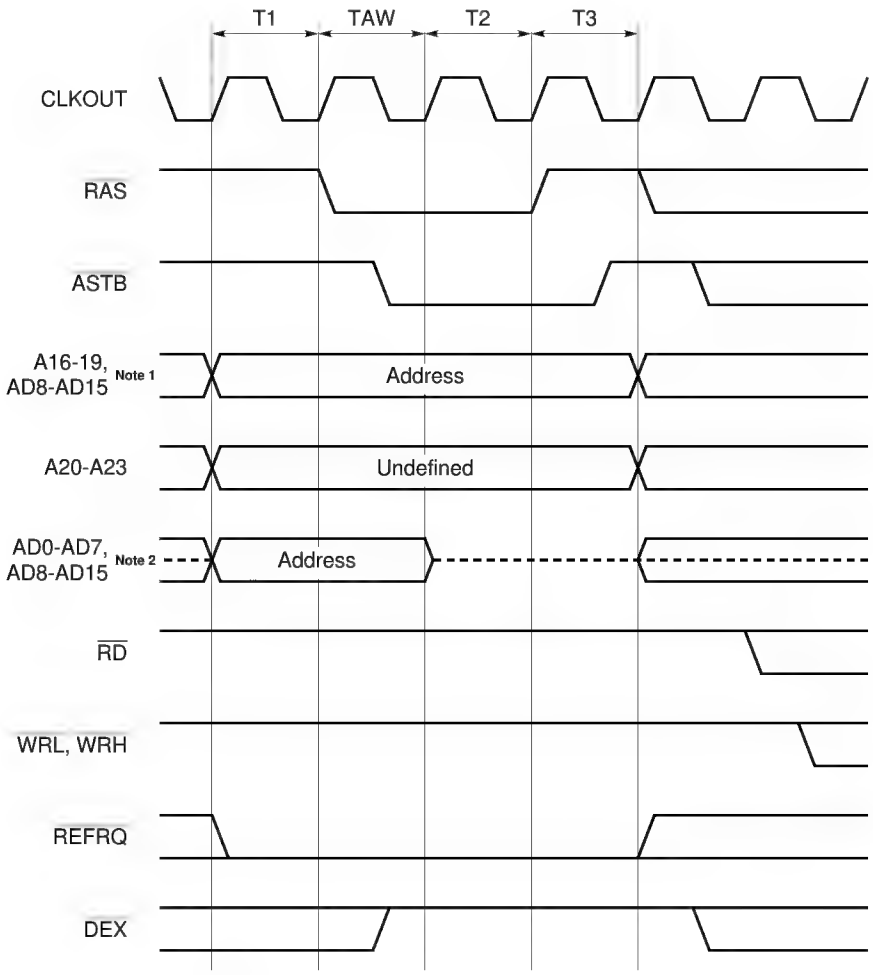
The RAS-only refresh cycle and CAS before RAS refresh cycle of DRAM are supported.

The $\overline{REFRQ}$ pin becomes active to signal a refresh cycle.

The maximum length of time until a refresh cycle is inserted ($\overline{RESET}\uparrow \rightarrow \overline{REFRQ}$) after initialization by reset is 1 μs + 13.5T (T: number of cycles when ϕ = fxx/8 mode).

Figure 4-6 shows the refresh cycle timings.

Figure 4-6. Refresh Cycle Timing Chart
(TAW is valid only when address wait is inserted)



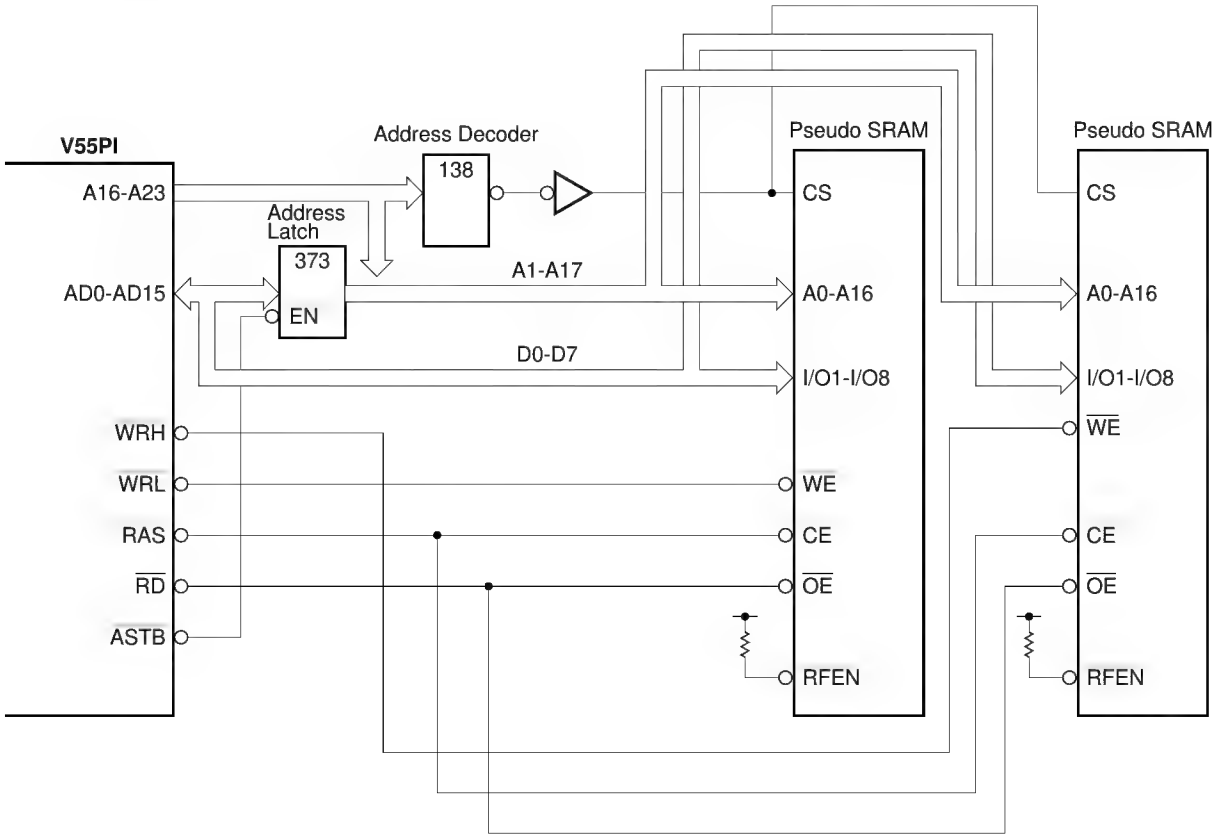
Notes 1. When external bus width 8 bits
2. When external bus width 16 bits

Remark Dotted line indicates high impedance.

4.7.5 Pseudo-SRAM connection

Figure 4-7 shows a circuit example of connecting a pseudo-SRAM.

Figure 4-7. Example of Pseudo-SRAM Connection Circuit (Address Wait Insertion)



Remark Number of address latches and address decoders depend on external memory sizes.

4.8 Bus Hold Function

The V55PI has a bus hold function. The fact that an external device wants to use the external bus is indicated by externally inputting a high-level signal to the HLDRQ pin. When the high-level HLDRQ signal is detected, the following operations are performed.

(1) The $\overline{\text{HLDAK}}$ signal is set to the low level (indicating that the bus has been released to the external device).
(2) Transition to the hold mode.
In the hold mode, the HLDRQ signal is always checked.
(3) The HLDRQ signal low level is detected.
(4) The $\overline{\text{HLDAK}}$ signal is set to the high level (indicating that the bus has not been released to the external device).
(5) The hold mode is released.
(6) Execution of the instruction is restarted.

A bus hold request is also acknowledged in the HALT mode. When the HLDRQ signal is set to the low level and the hold mode is released, the mode returns to the HALT mode. The bus hold request is not acknowledged in the following cases.

- In STOP mode
- In execution of one instruction following the BUSLOCK prefix instruction

In the hold mode, a refresh cycle can be inserted by specification by means of the refresh mode register (RFM).
First, the $\overline{\text{HLDAK}}$ signal should forcibly be set to the high level for each refresh timing. After confirming that the HLDRQ signal has been set to the low level, a refresh cycle is generated. After the $\overline{\text{HLDAK}}$ signal has been set in to the low level again, the following operations are performed.

- When the HLDRQ signal is set to the high level : Transition to the hold mode again
- When the HLDRQ signal remains low : The hold mode is released and execution of the instruction restarts.

★ **Caution If the HLDRQ signal becomes active during reset, reset vector read after reset is released is not executed, and transition to the hold mode occurs.**

4.9 Bus Mastership

The V55PI bus mastership priority is as follows.

During execution of the interrupt acknowledge cycle[Note], the refresh cycle, the transition to hold mode, and the DMA transfer cycle are held pending temporarily. However, in the STOP mode, no bus cycle is generated.

Note Preprocessing period from interrupt request acknowledgment by the CPU until the branch to the interrupt service routine.

(1) HOLD mode

The HOLD mode is set except during execution of one instruction following the BUSLOCK prefix and the interrupt acknowledge cycle.

(2) Refresh cycle

If refresh cycle insertion has been enabled, a refresh cycle is generated. However, the $\overline{\text{HLDAK}}$ signal is forcibly set to the high level in the HOLD mode. When the HLDRQ signal is set to the low level, the refresh cycle is executed (If refresh cycle automatic insertion is enabled during HOLD mode).

(3) DMA transfer cycle

DMA transfer bus cycle by incorporated DMA controller.

(4) Other bus cycle

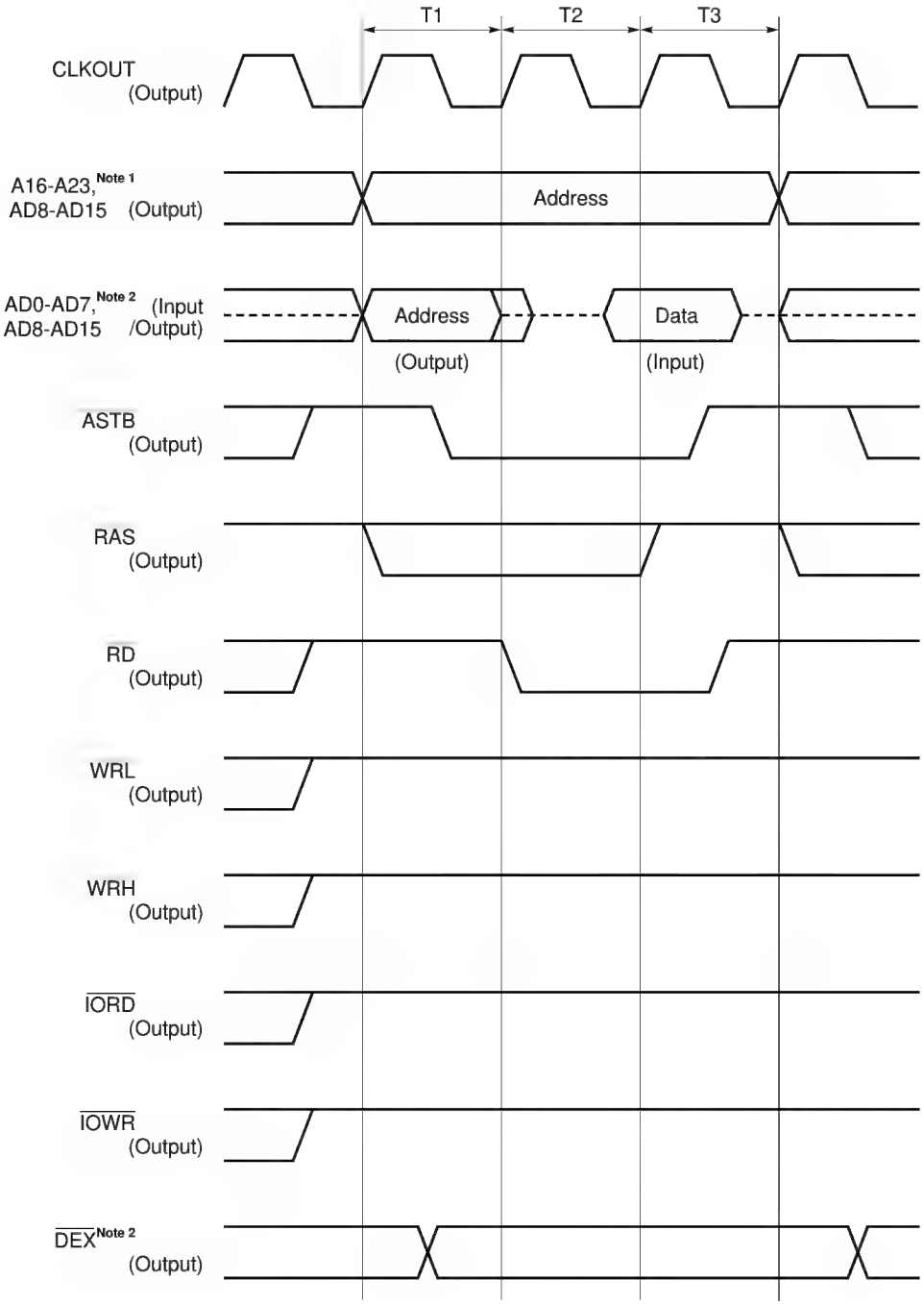
This is a V55PI memory manipulate instruction execution and program code fetch operation bus cycle.

4.10 Bus Timings

Figures 4-8 to 4-19 show main bus timings (except the refresh and DMA transfer timing).

While no bus access is made, the control pin remains inactive and address/data bus output is undefined.

Figure 4-8. Memory Read Cycle

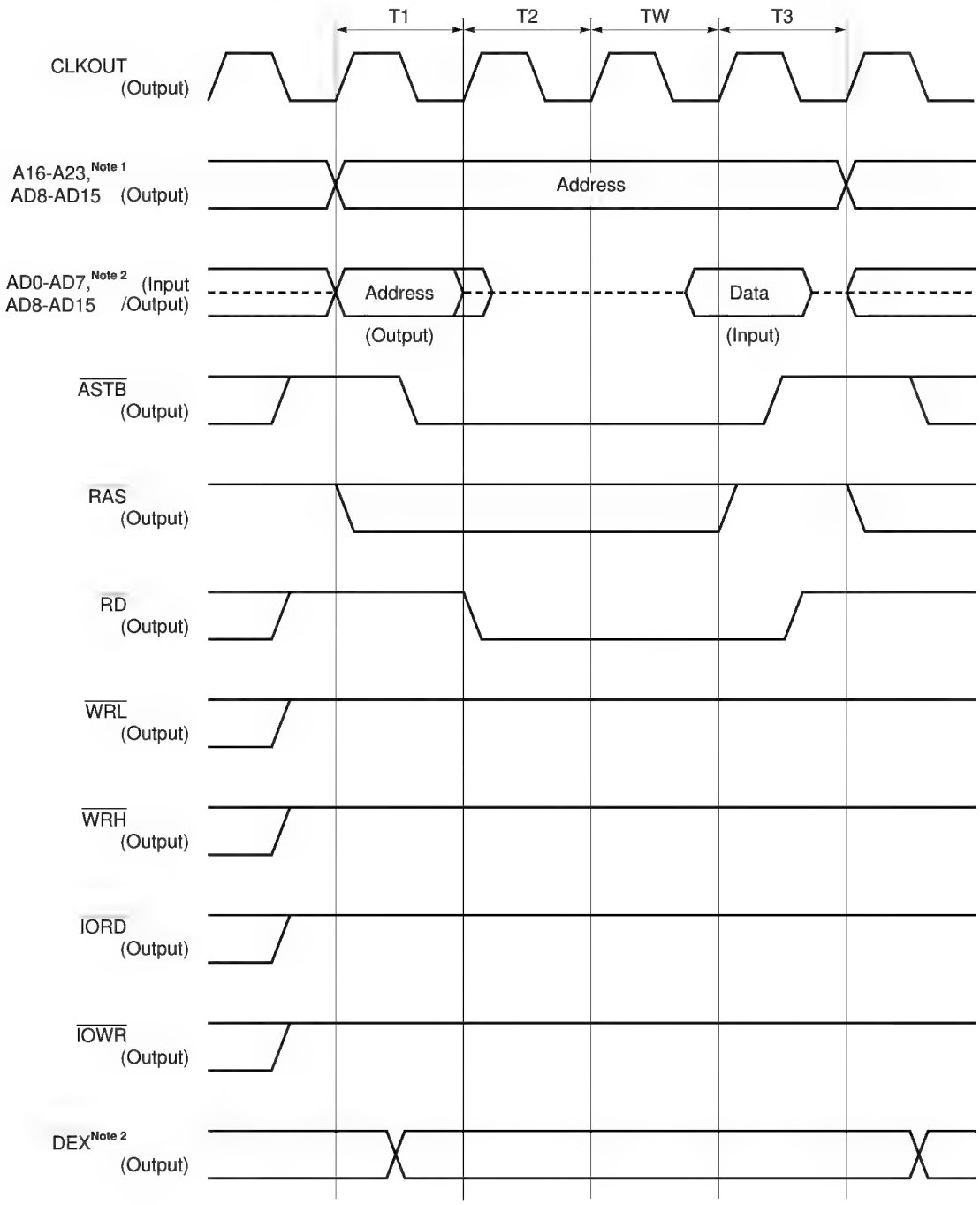


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-9. Memory Read Cycle with Data Wait (1 Wait) Inserted

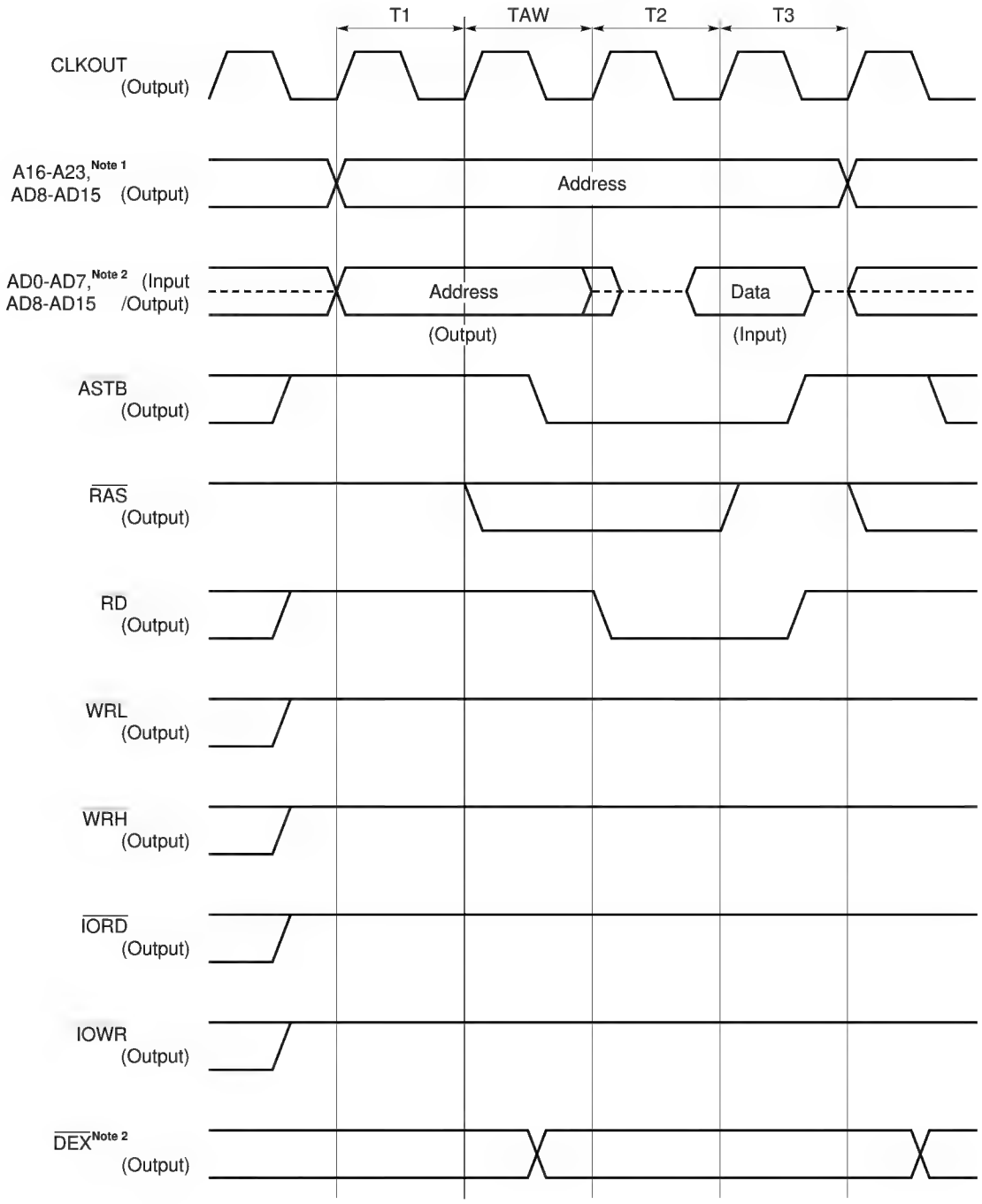


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-10. Memory Read Cycle with Address Wait Inserted

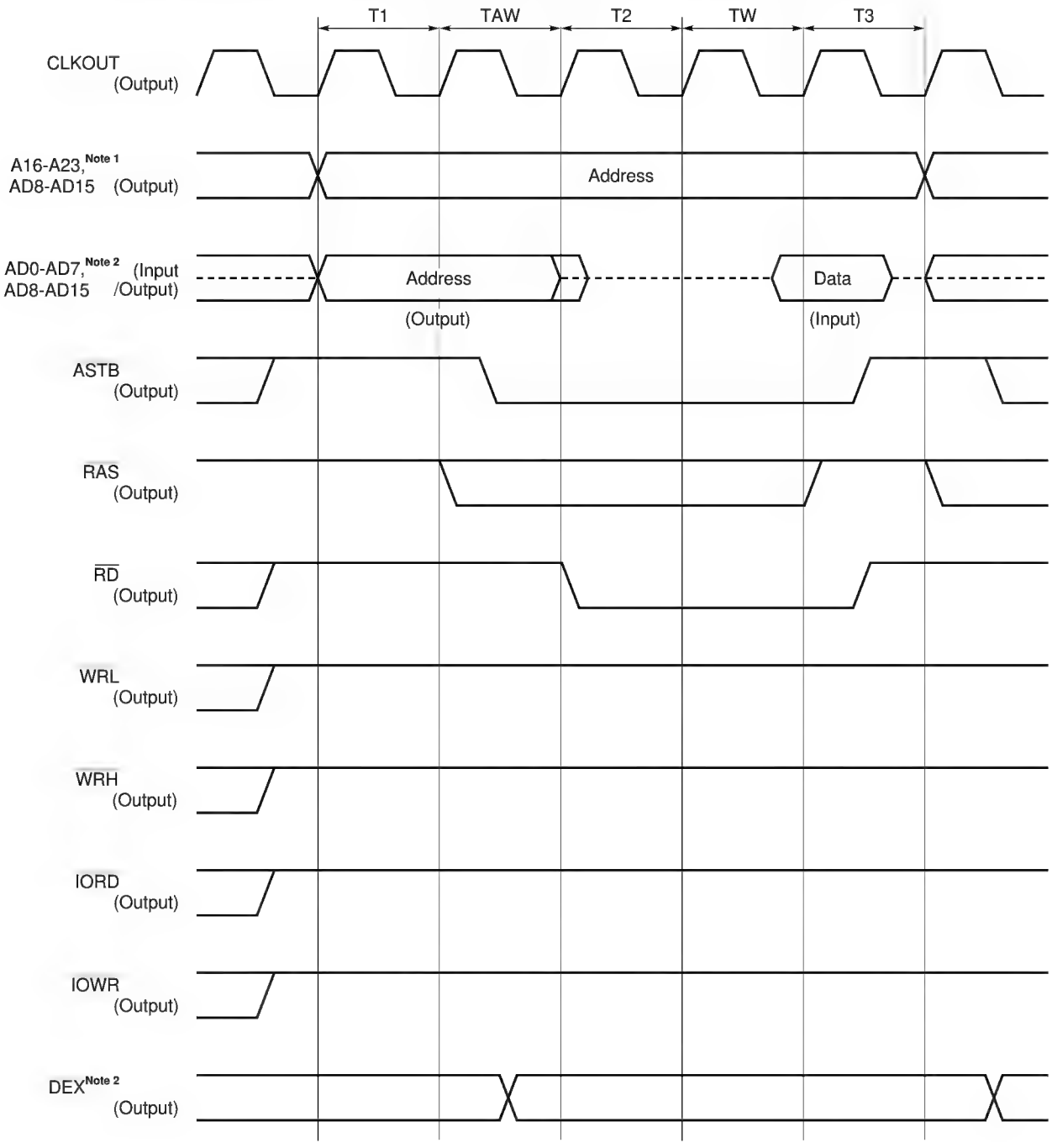


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-11. Memory Read Cycle with Data Wait (1 Wait) and Address Wait Inserted

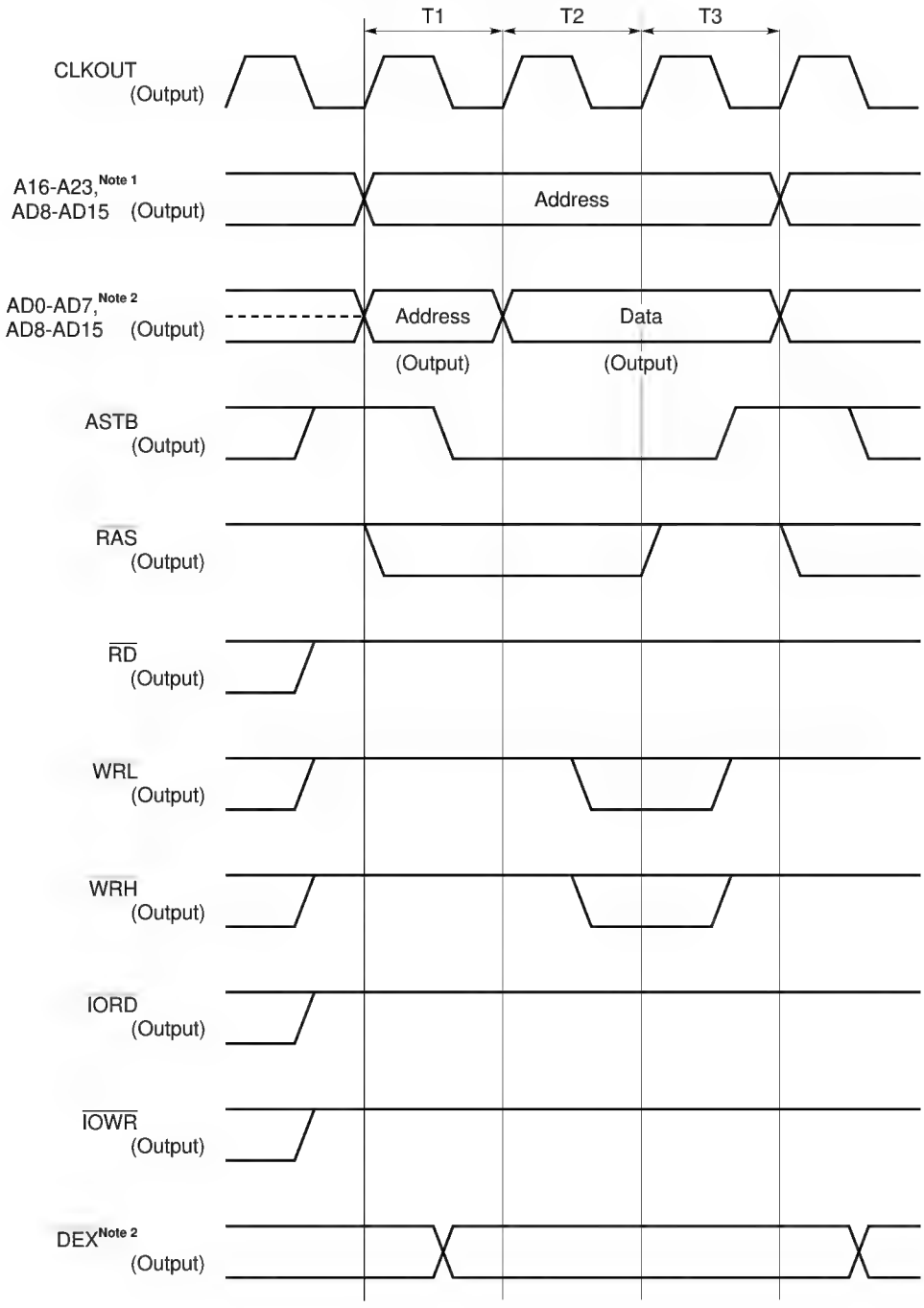


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-12. Memory Write Cycle

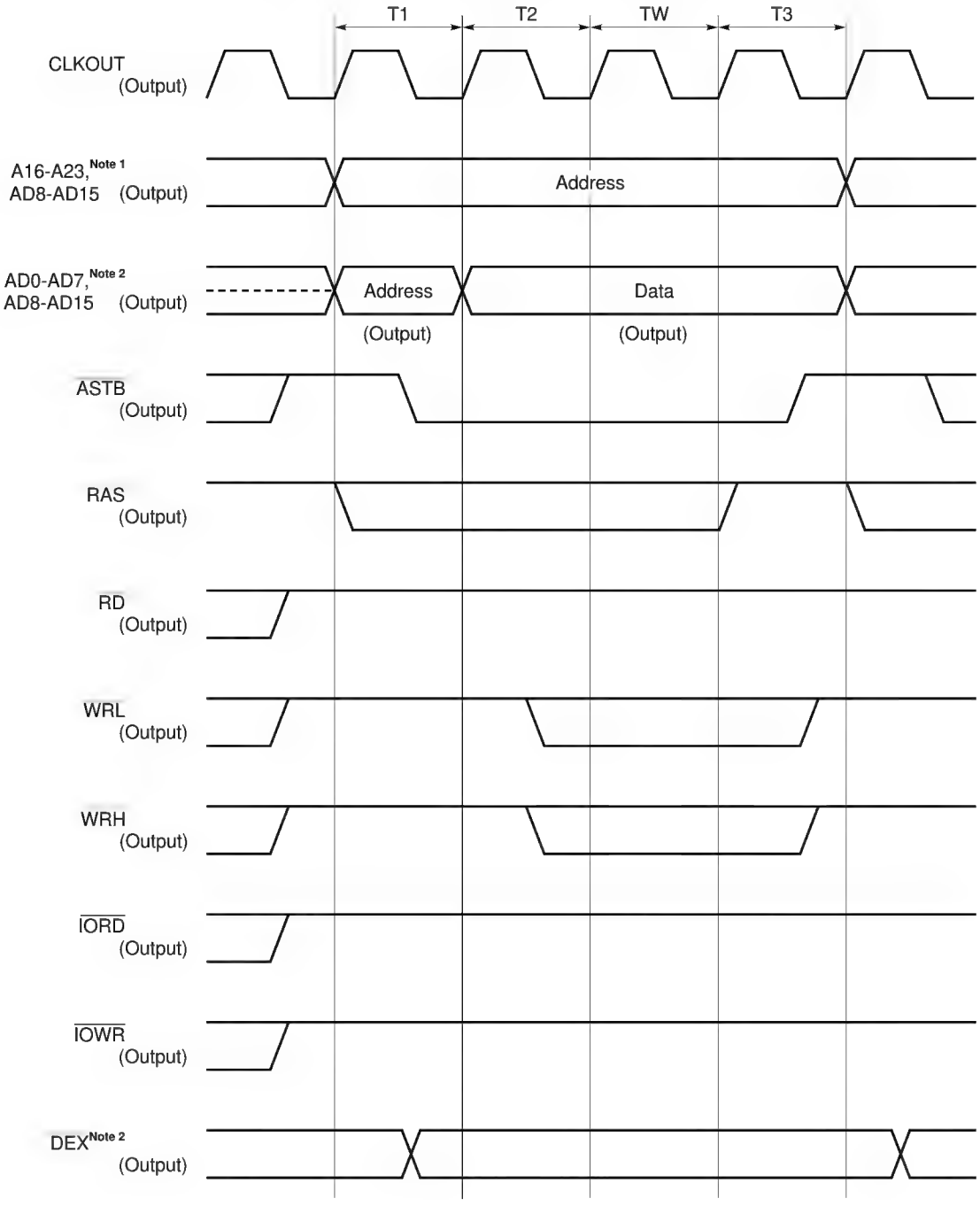


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-13. Memory Write Cycle with Data Wait (1 Wait) Inserted

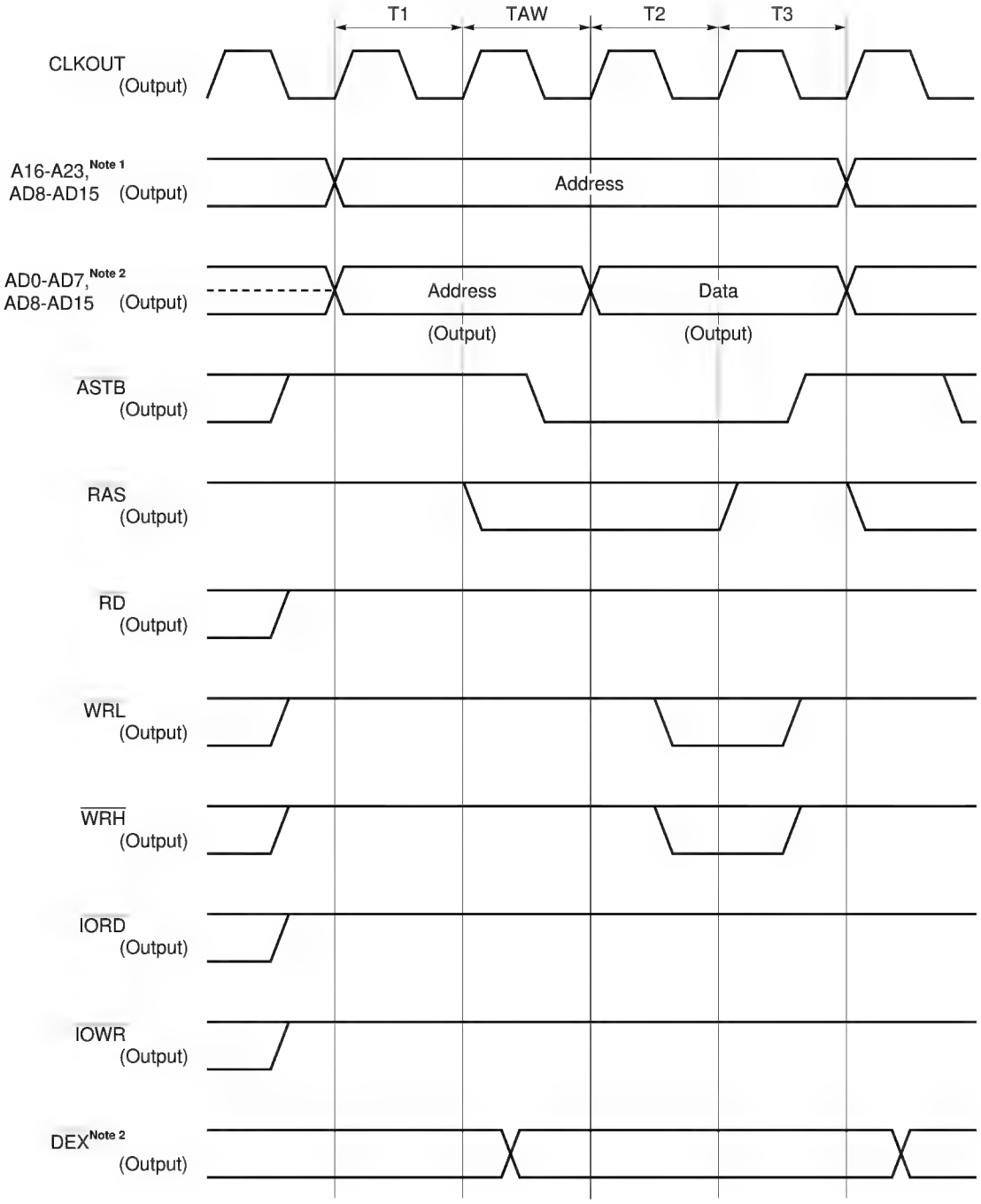


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-14. Memory Write Cycle with Address Wait Inserted

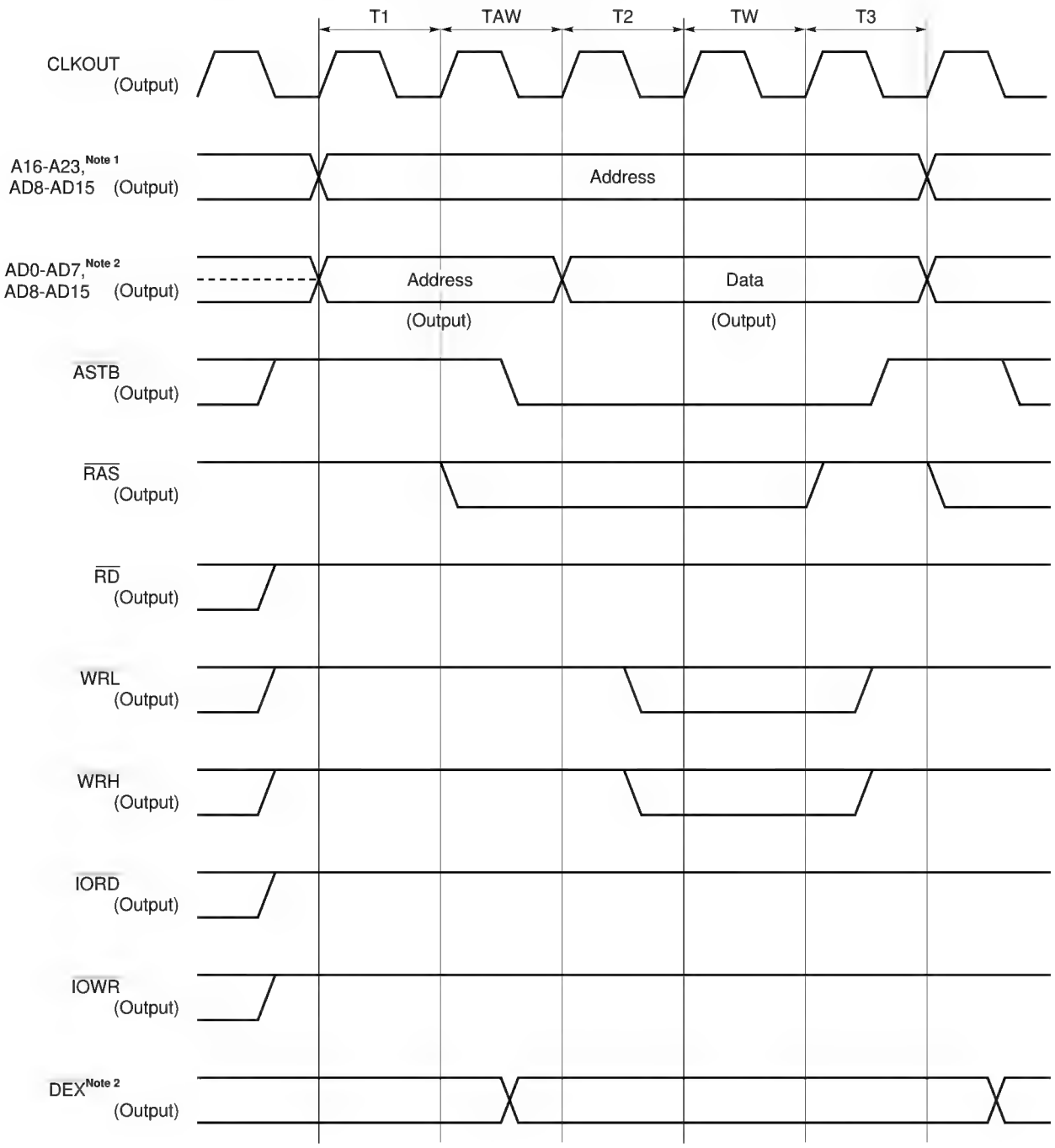


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-15. Memory Write Cycle with Data Wait (1 Wait) and Address Wait Inserted

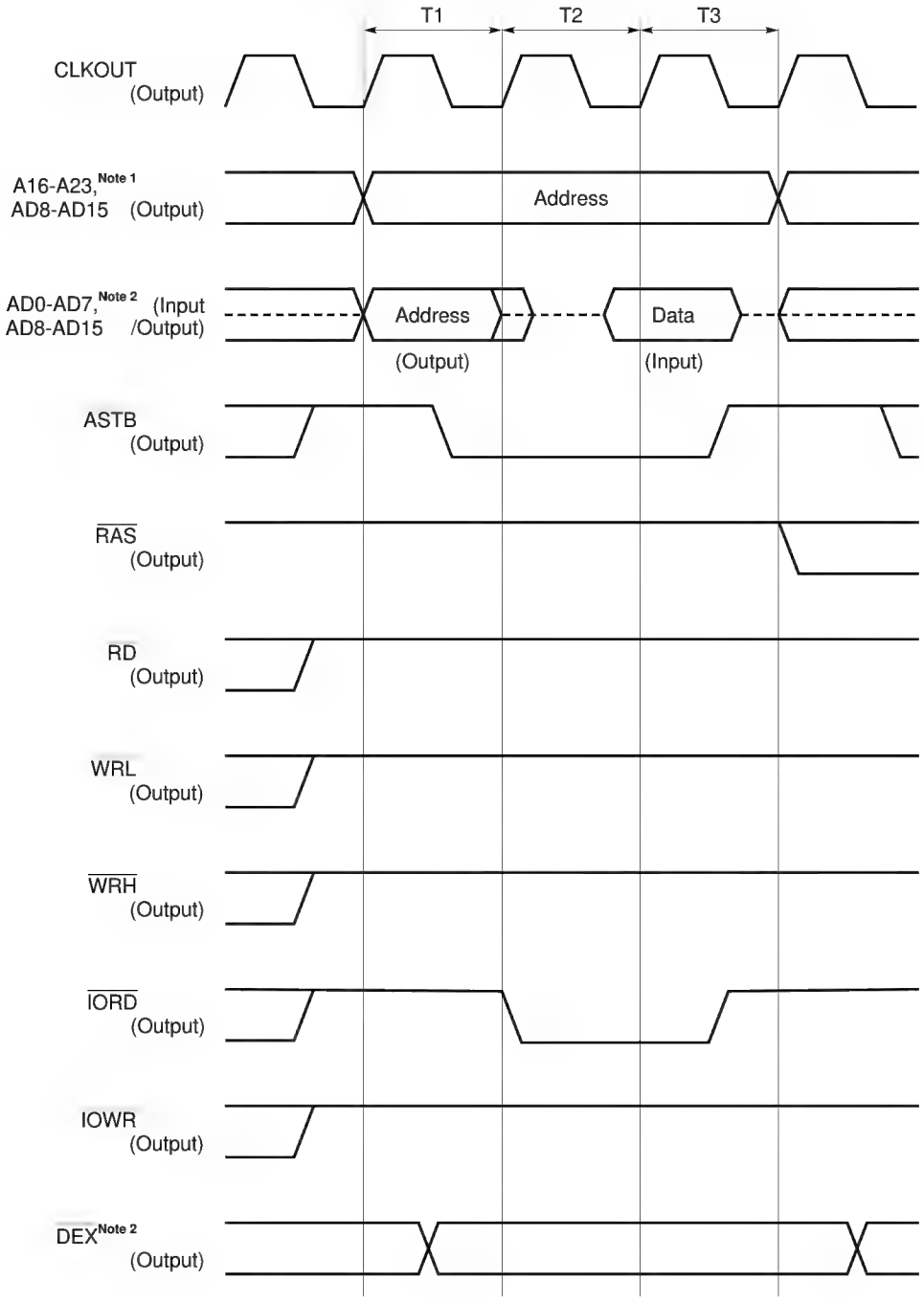


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-16. I/O Read Cycle

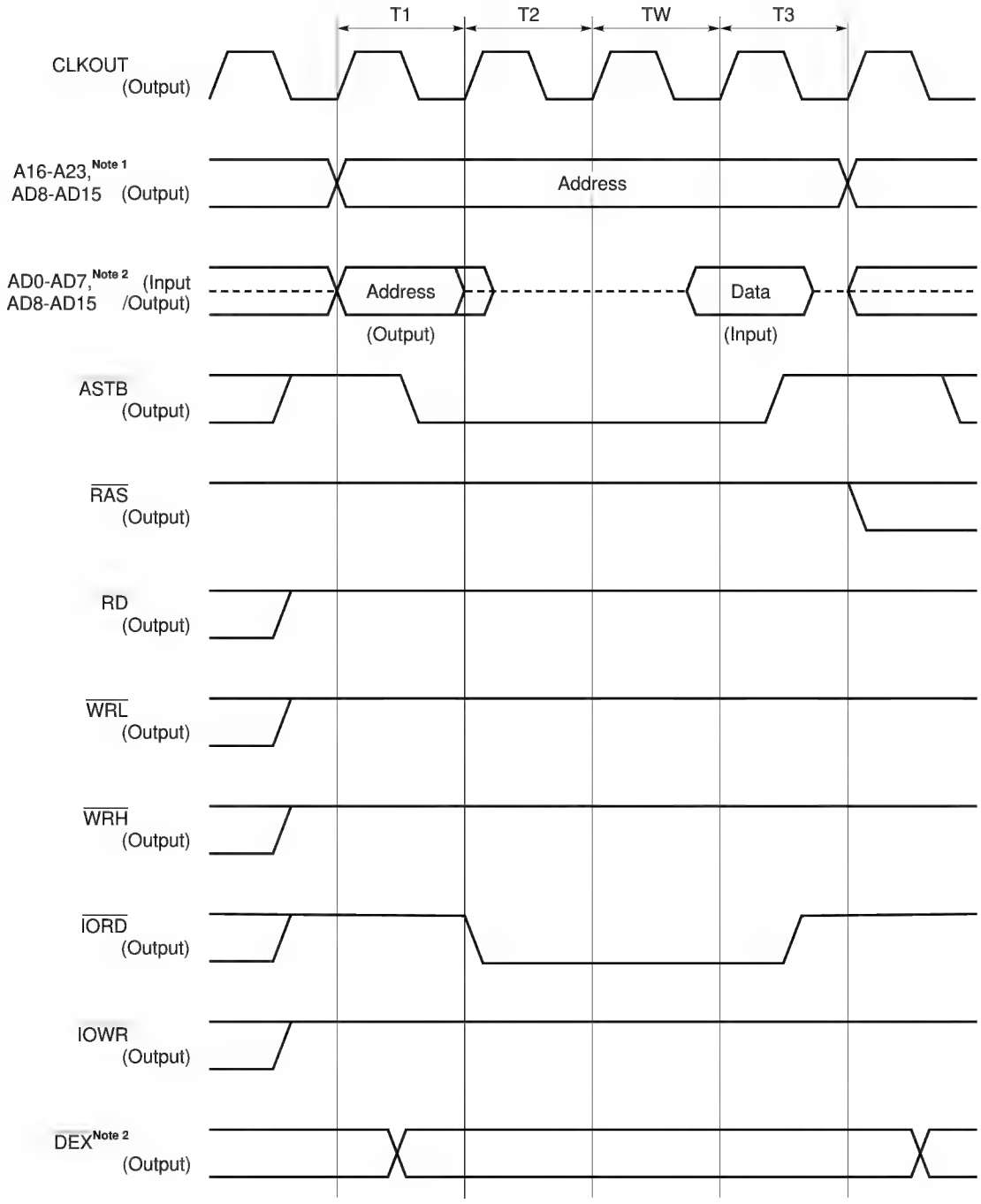


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-17. I/O Read Cycle with Data Wait (1 Wait) Inserted

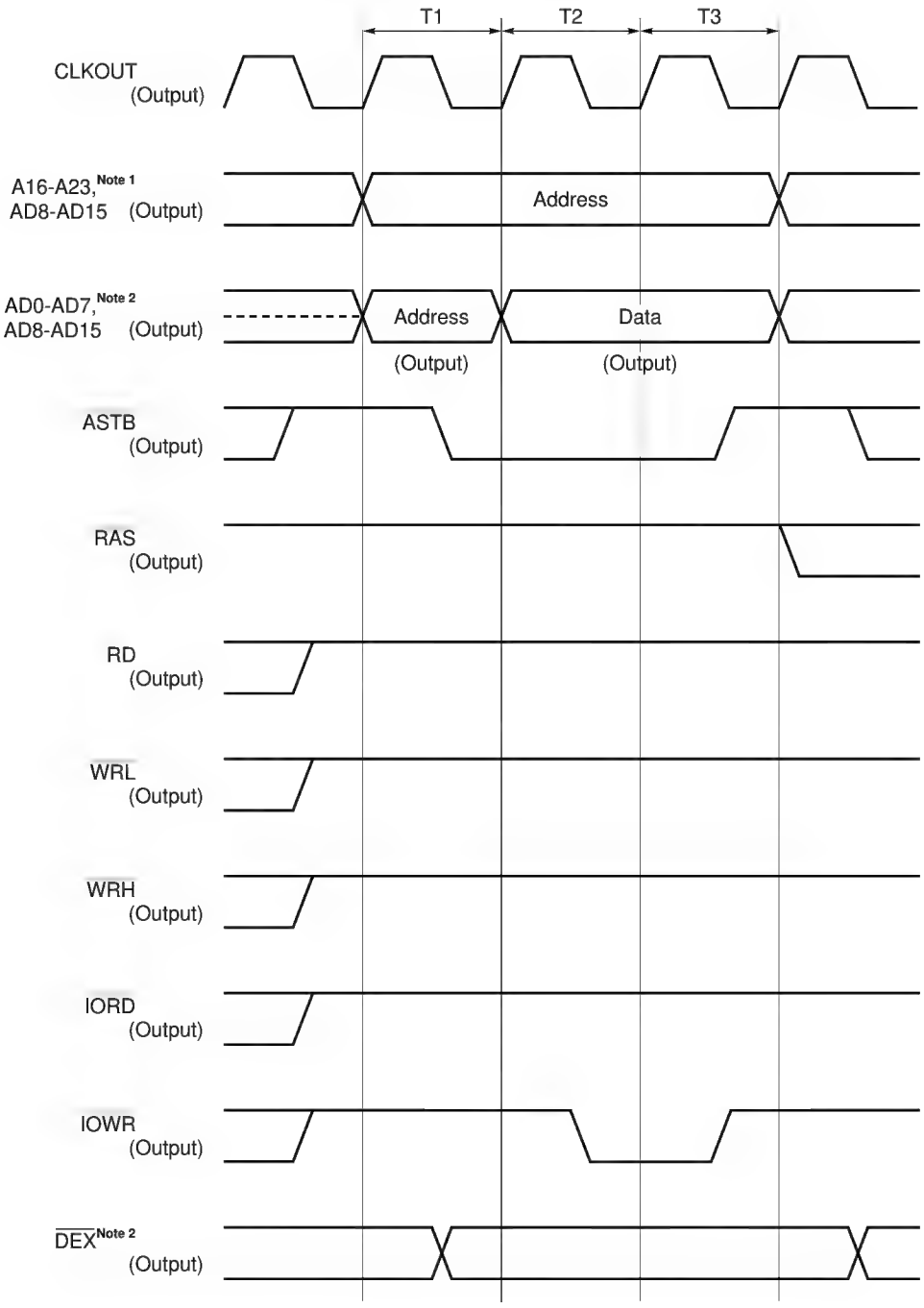


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-18. I/O Write Cycle

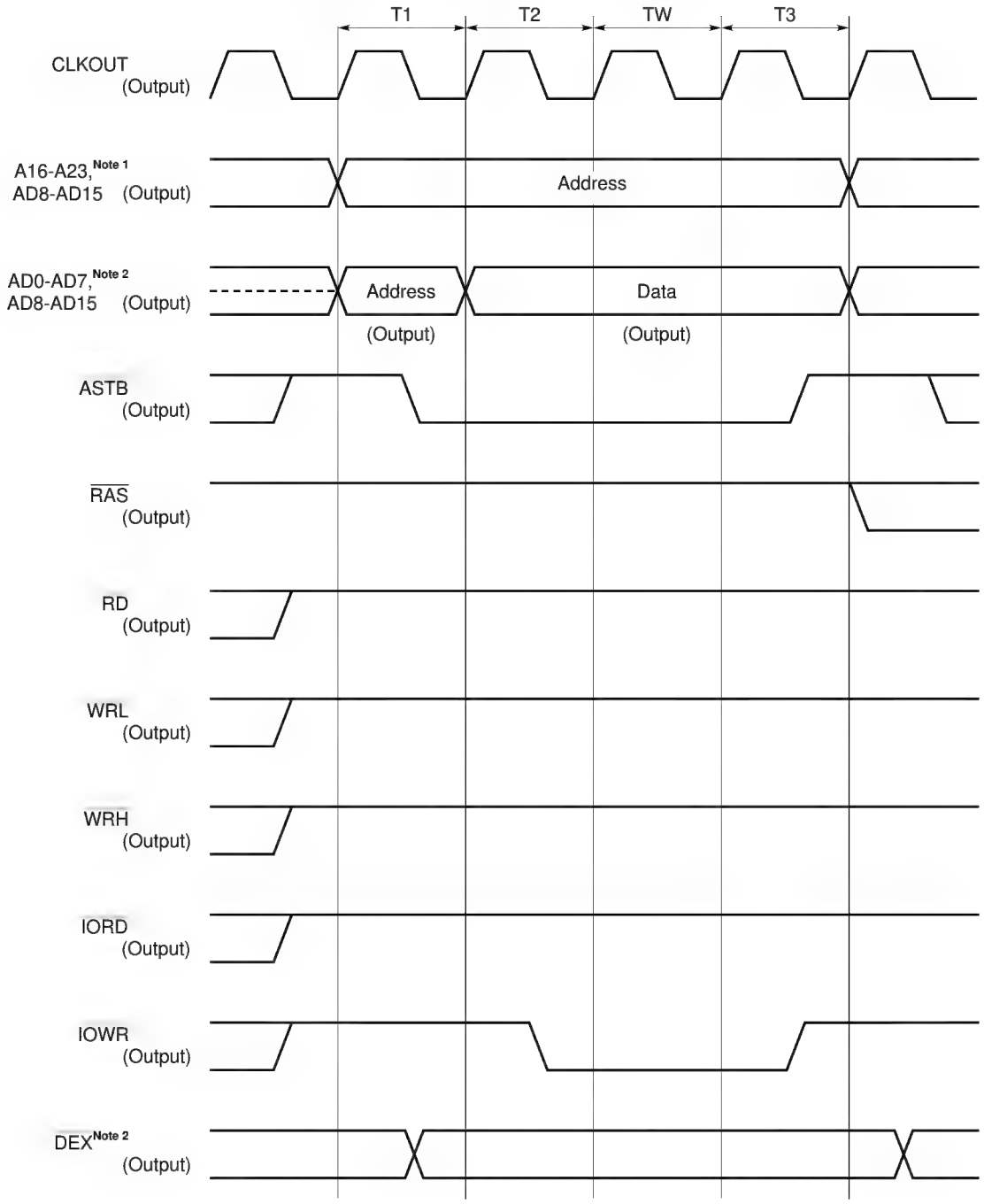


Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

Figure 4-19. I/O Write Cycle with Data Wait (1 Wait) Inserted



Notes 1. When external bus width 8 bits

2. When external bus width 16 bits

Remark Dotted line indicates high-impedance.

[MEMO]

CHAPTER 5 INTERRUPT FUNCTIONS

The V55PI incorporates a powerful interrupt controller (INTC) which controls for a total of 25 maskable hardware interrupt requests: 19 internal and 6 external. The interrupt controller controls multiple-interrupt servicing based on the programmable priority. Interrupt service modes provided comprise a vectored interrupt function, macro service function, and register bank switching function.

5.1 Features

V55PI interrupt functions offer the following features:

- Comprehensive servicing modes for interrupt requests

  – Vectored interrupt function: Branch to the interrupt service routine specified by the vector table
  – Register bank switching function: High-speed interrupt response by automatic register bank switching
  – Macro service function: High-speed interrupt servicing by microprogram (firmware)

- 4-level programmable priority control
- Priority-based multiple-interrupt servicing control
- Variety of macro service functions (following 7 modes) closely tied to V55PI on-chip peripheral hardware

  EVTCNT : Event count process
  BLKTRS : Data transfer between a special function register and an external memory buffer
  BLKTRS-C : Data transfer between a special function register and an external memory buffer (with transfer data detection function)
  DTACMP : Special function register status detection
  DTADIF : Time measurement using timer capture function
  RTOPTRN : Automatic control of real-time output port
  DTACMP-M : Data transfer between an external I/O and memory

- 7 external interrupt request inputs (NMI, INTP0 to INTP5)
- Maskable interrupt requests are individually maskable.

Control of the operation of the interrupt controller which includes these features is performed by the interrupt mode control register (IMC), interrupt mask flag registers (MK0, MK1), external interrupt mode registers (INTM0, INTM1), and interrupt request control registers (IC××) provided for each maskable interrupt source.

Interrupt control instructions include the EI and DI instructions which control interrupt enabling and disabling, the RETI and RETRBI instructions which direct the return from an interrupt, and the FINT instruction which notifies the on-chip interrupt controller of the end of interrupt servicing.

5.2 Interrupt Sources

Interrupt requests handled by the V55PI are divided into the following 4 types:

(1) Non-maskable interrupt requests (NMI, watchdog timer)
(2) Maskable interrupt requests (interrupt requests from on-chip peripheral circuits and external pins)
(3) Software interrupt requests (BRK and CHKIND instructions, division errors, etc.)
(4) Exception trap (special write error)

A list of interrupt sources is given in Table 5-1.

A total of 27 interrupt sources, non-maskable and maskable, are handled by the interrupt controller. Software interrupts in which an interrupt is generated by execution of an instruction and the exception trap (special write error) are not subject to control by the interrupt controller.

Table 5-1. Interrupt Sources (1/2)

Interrupt Type	Default Priority	Interrupt Request Signal	Interrupt Source			Default Vector Table Number	Vector Address	Macro Service	Register Bank Switching	Macro Service Control Word Address
			Interrupt Request Control Register	Generation Source	Generating Unit					
Non-maskable	1	NMI	—	NMI pin input	—	2	00008H	No	No	—
	2	INTWDT		Watchdog timer overflow	WDT	8	00 × 20H	No	No	
Maskable	3	INTP0	IC9	INTP0 pin input	External	9	00 × 24H	Yes	Yes	012H
	4	INTP1	IC10	INTP1 pin input		10	00 × 28H	Yes	Yes	014H
	5	INTP2	IC11	INTP2 pin input		11	00 × 2CH	Yes	Yes	016H
	6	INTP3	IC12	INTP3 pin input		12	00 × 30H	Yes	Yes	018H
	7	INTP4	IC13	INTP4 pin input		13	00 × 34H	Yes	Yes	01AH
	8	INTP5	IC14	INTP5 pin input		14	00 × 38H	Yes	Yes	01CH
	9	INTCM00	IC16	CM00 match detection	Timer	16	00 × 40H	Yes	Yes	020H
	10	INTCM01	IC17	CM01 match detection		17	00 × 44H	Yes	Yes	022H
	11	INTCM10	IC18	CM10 match detection		18	00 × 48H	Yes	Yes	024H
	12	INTCM11	IC19	CM11 match detection		19	00 × 4CH	Yes	Yes	026H
	13	INTCM21	IC20	CM21 match detection		20	00 × 50H	Yes	Yes	028H
	14	INTCM31	IC21	CM31 match detection		21	00 × 54H	Yes	Yes	02AH
	15	INTD0	IC22	DMA channel 0_main	DMA	22	00 × 58H	Yes	Yes	02CH
	16	INTD0S	IC23	DMA channel 0_sub		23	00 × 5CH	Yes	Yes	02EH
	17	INTD1	IC24	DMA channel 1_main		24	00 × 60H	Yes	Yes	030H
	18	INTD1S	IC25	DMA channel 1_sub		25	00 × 64H	Yes	Yes	032H

Table 5-1. Interrupt Sources (2/2)

Interrupt Type	Default Priority	Interrupt Request Signal	Interrupt Source			Default Vector Table Number	Vector Address	Macro Service	Register Bank Switching	Macro Service Control Word Address
			Interrupt Request Control Register	Generation Source	Generating Unit					
Maskable	19	INTSER0	IC26	UART reception error (ch0)	Serial I/F	26	00 × 68H	No	Yes	—
	20	INTSER1	IC27	UART reception error (ch1)		27	00 × 6CH	No	Yes	—
	21	INTSR0/ INTCSI0	IC28	UART reception (ch0)/ serial transmission/reception (ch0)		28	00 × 70H	Yes	Yes	038H
								Yes	Yes	
	22	INTSR1/ INTCSI1	IC29	UART reception (ch1)/ serial transmission/reception (ch1)		29	00 × 74H	Yes	Yes	03AH
								Yes	Yes	
	23	INTST0	IC30	UART transmission (ch0)		30	00 × 78H	Yes	Yes	03CH
	24	INTST1	IC31	UART transmission (ch1)		31	00 × 7CH	Yes	Yes	03EH
	25	INTSIT	IC32	STM match detection	SIT	32	00 × 80H	No	Yes	—
	26	INTPAI	IC36	Parallel I/F	Parallel I/F	36	00 × 90H	Yes	Yes	008H
	27	INTAD	IC37	A/D converter	A/D converter	37	00 × 94H	Yes	Yes	00AH
Software	—	—	—	Divide error	—	0	00000H	No	No	—
				BRK flag (single step)		1	00004H	No	No	
				BRK3 instruction		3	0000CH	No	No	
				BRKV instruction		4	00010H	No	No	
				CHKIND instruction		5	00014H	No	No	
				Input/output instruction (IBRK flag)		6	00018H	No	No	
				BRK imm8		Note	00 × NoteH	No	No	
				BRKCS instruction		—	—	No	Yes	
Exception trap				FPO instruction/ exception trap		7	0001CH	No	No	

Note Indicates that the value is variable in the range 0 to 255 (0 to FFH).

Remark "×" indicates that the value is determined by the V0 and V1 bits of the IMC register.

5.2.1 Non-maskable interrupt

A non-maskable interrupt request is always acknowledged with highest priority even in the interrupt disabled state (IE = 0) irrespective of the priority specified by software. There are two sources: detection of external valid edge input to the P10 pin (NMI) and internal watchdog timer overflow signal output (INTWDT).

Unlike maskable interrupt requests, these interrupt requests are not subject to programmable priority control. Because NMI and INTWDT are not subject to multiple-interrupt service control, nesting is possible at any level.

Non-maskable interrupt requests are serviced by normal vectored interrupts. As default priorities, NMI is ranked higher than INTWDT. When NMI and INTWDT requests are simultaneously generated, the NMI is acknowledged first.

(1) NMI

NMI is edge-detected. The edge direction can be selected by ES0 (bit 0) of the external interrupt mode register 0 (INTM0), one of the interrupt control registers. An interrupt is generated by a falling edge when ES0 = 0, and by a rising edge when ES0 = 1.

Only a vector response is possible for this interrupt, and the vector number is fixed at 2.

When NMI is acknowledged, the interrupt disabled state is set (IE = 0) and maskable interrupts are disabled (however, an interrupt by means of macro service can be acknowledged: See **5.3.2 Priority control for simultaneously generated interrupts** and **5.12 Examples of Interrupt Request Service Operations**).

If a valid edge is input again to the NMI pin during NMI processing, the current NMI processing is interrupted, and the NMI processing for that valid edge is begun.

Caution In the following text, "NMI" is not used as a generic term for non-maskable interrupts, but to indicate the detection of valid edge input to the P10 pin.

(2) INTWDT

INTWDT is an interrupt for the detection of inadvertent program looping, system errors, etc. This interrupt is serviced only as a vectored interrupt, and its initial vector number is fixed at 8 (however, the vector address can be changed by bits V0 and V1 of the interrupt mode control register (IMC)).

When an interrupt is acknowledged, the interrupt disabled state is set (IE = 0) and other interrupts are disabled (however, a macro service and NMI can be acknowledged: See **5.3.2 Priority control for simultaneously generated interrupts**).

See **CHAPTER 12 WATCHDOG TIMER FUNCTION** for further details.

5.2.2 Maskable interrupts

The V55PI has a total of 25 maskable interrupt sources, 6 external and 19 internal, with multiple-interrupt service control performed by the interrupt controller.

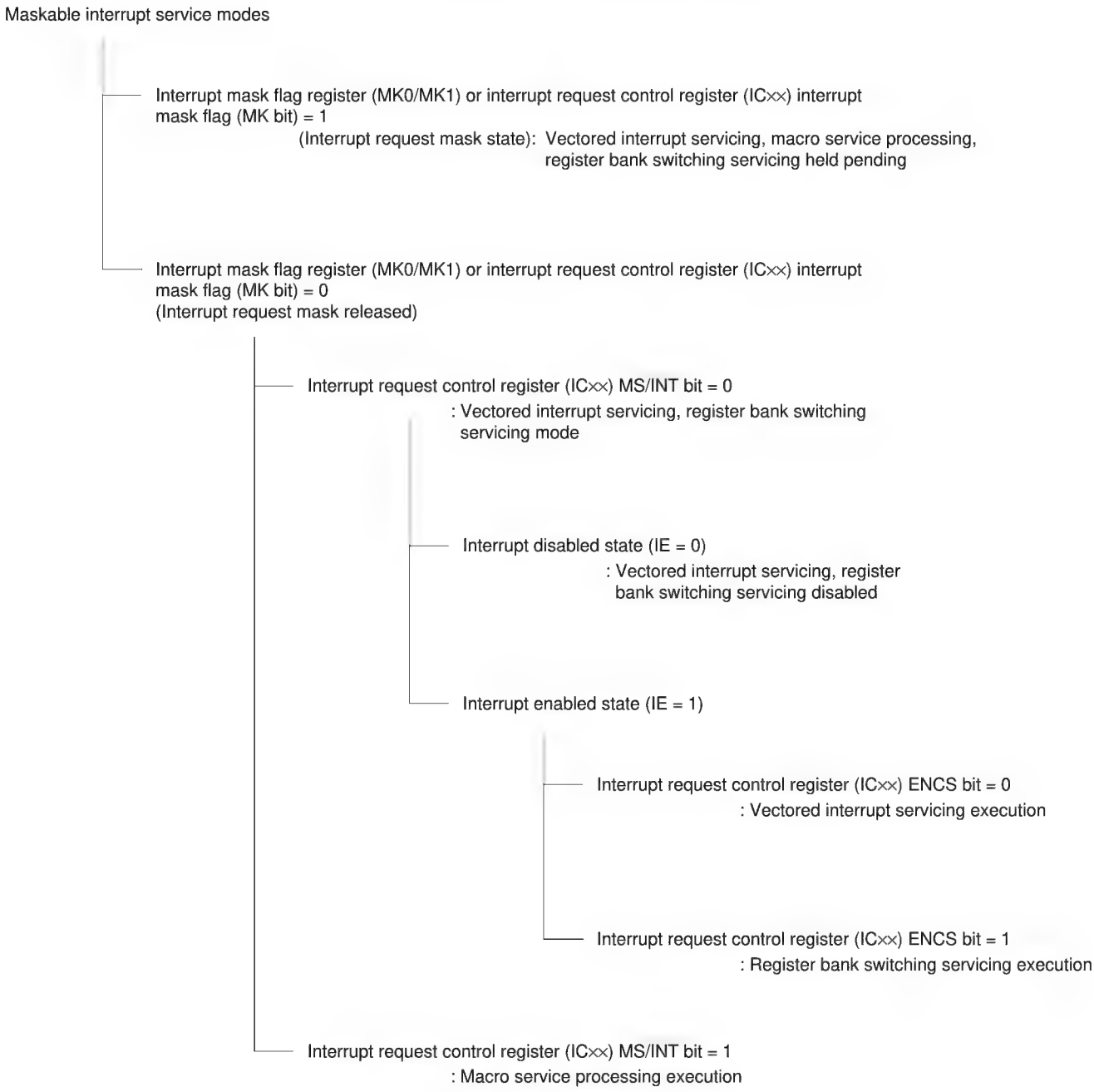
There is a mask bit in the interrupt mask flag register (MK0/MK1) and the per-source interrupt request control registers (ICxx), enabling interrupt requests to be masked as required.

Any of the following service methods can be specified for maskable interrupts acknowledged by means of the interrupt request control register (ICxx):

<1> Vectored interrupt
<2> Macro service processing
<3> Register bank switching function

Bit 9 (interrupt enable flag: IE flag) of the PSW controls enabling/disabling of all maskable interrupt servicing.
IE = 0: All maskable interrupts disabled.
IE = 1: Maskable interrupt request acknowledgment enabled.
The IE flag is reset (0) when an interrupt request is acknowledged (it is not changed by macro service).
Maskable interrupt service modes are shown in Figure 5-1.

Figure 5-1. Maskable Interrupt Service Modes



5.3 Priority Control

Priority control for multiple interrupts is performed for maskable interrupts, and one of four priority levels can be set arbitrarily for each interrupt source.

The V55PI's on-chip interrupt controller performs priority control on the basis of this information.

The basic control concept for multiple interrupts service control is identical to that of the V25 and V35.

5.3.1 Priority control for multiple interrupts

Priority control for multiple interrupt service is a function whereby, if there is an interrupt request of a higher level than that of the interrupt currently being serviced, the interrupt currently being serviced is discontinued and acknowledgment performed, and servicing of the higher-priority interrupt is performed.

If the priority level of an interrupt is the same as or lower than that of the interrupt currently being serviced, that interrupt request is held pending.

Multiple interrupt service priority control is performed in the interrupt enabled state (IE = 1). Therefore, the interrupt enabled state (IE = 1) must be set in the interrupt service routine when multiple interrupt servicing is to be performed.

Any of four multiple interrupt service control priority levels between 0 and 3 (0 being the highest level) can be set for each maskable interrupt request by software. Priority level setting is performed by bits PR00 and PR01 of the interrupt request control register (IC××) provided for each maskable interrupt request. When a reset is performed, interrupt requests are masked by the MK bit and the priority is set to level 3 by bits PR00 and PR01.

Maskable Interrupt Priority

(High) Level 0 > Level 1 > Level 2 > Level 3 (Low)

In a maskable interrupt response after an interrupt is acknowledged, an FINT instruction must be executed directly before the return instruction from interrupt service (RETI instruction or RETRBI instruction) to indicate to the interrupt controller that servicing has ended at the end of the interrupt service routine. If an FINT instruction is not executed, subsequent priority control cannot be performed correctly.

Interrupt servicing suspended by multiple service control is resumed when servicing of the higher-priority-level interrupt ends and the FINT and RETI or RETRBI instructions have been executed.

A pending interrupt request is acknowledged after execution of the FINT and RETI or RETRBI instructions when the interrupt servicing currently being executed is completed.

Cautions 1. Since NMI and INTWDT interrupts are not subject to multiple interrupt control, they can be acknowledged at any time. Also, although the interrupt disabled state (IE = 0) is set after an NMI or INTWDT interrupt request is acknowledged, if the interrupt enabled state (IE = 1) is set in the interrupt service routine, multiple maskable interrupt servicing is executed.

2. The above priority control has no effect on the interrupt request for which a macro service response is set. Only the default priority is valid. However, the macro service completion interrupt is affected by the priority control.

As one macro service interrupt is executed in the same way as one instruction, other interrupts (including macro service) are not acknowledged during the operation. (Refer to 5.4.3. (2) Macro service completion interrupt).

5.3.2 Priority control for simultaneously generated interrupts

When multiple interrupt requests are generated simultaneously, the following priority order applies.

Priority Order of Simultaneously Generated Interrupt Requests

(High) NMI > INTWDT > maskable interrupt serviced by macro service
> maskable interrupt serviced by vectored interrupt, or register bank switching function (Low)

Also, between multiple maskable interrupts serviced by a vectored interrupt or the register bank switching function, interrupt requests are acknowledged sequentially according to the priority specified by bits PR00 and PR01 of each interrupt request control register (TC××). When there are multiple interrupts with a certain priority, interrupt requests are acknowledged according to the default priority indicated by Table 5-1.

See **5.12 Examples of Interrupt Request Service Operations** for concrete examples.

(1) Macro service and NMI execution order

Because a macro service interrupt is acknowledged independently of whether the DI or EI state is in effect, macro service is also executed while an NMI or INTWDT interrupt service routine is executed.

If an NMI request and macro service request are generated simultaneously, the NMI request is acknowledged first under the servicing control exercised by the interrupt controller. However, note that because a macro service is executed immediately after a branch to the NMI service routine, the program execution order is as follows when the interrupt service routine is considered as a unit.

Macro service → NMI

To disable macro service execution in an NMI (INTWDT) service routine, it is necessary to manipulate the interrupt mask flag register in the first instruction of the NMI service routine in order to mask macro service activation interrupt requests.

(2) NMI and INTWDT execution order

NMI and INTWDT requests are acknowledged independently of whether the DI or EI state is in effect. If NMI and INTWDT requests are generated simultaneously, service control is performed by the interrupt controller in accordance with the default priority order as follows:

NMI → INTWDT

However, because the INTWDT request is acknowledged immediately after a branch to the NMI service routine if the interrupt service routine is considered as a unit, note that the program execution order is as follows:

INTWDT → NMI

To disable INTWDT execution in an NMI service routine, it is necessary to execute a watchdog timer counter clearing instruction (RSTWDT) as the first instruction of the NMI service routine.

Even if an INTWDT request is generated at the same time as an NMI request, the watchdog timer interrupt is canceled and an interrupt is not executed (it is not, however, possible to disable NMI generation in the watchdog timer interrupt service routine).

5.3.3 Multiple interrupt control mode selection

Setting the bit 7 (PRSL) of the interrupt mode control register (IMC) allows the operating mode of the multiple interrupt servicing control to be selected for maskable interrupts of the lowest priority level (level 3). (See **5.5.1 Interrupt mode control register (IMC)**.)

PRSL bit = 1:

Multiple interrupt servicing control is performed for level 3 maskable interrupts.

For the multiple interrupt servicing control during the level 3 interrupt servicing, only an interrupt request of higher level (level 2 or higher) can be acknowledged, whereas multiple interrupt of the same level (level 3) is disabled and the interrupt is held pending.

This mode is selected when lowest-level (level 3) multiple interrupt servicing control is performed by interrupt priority control (interrupt request control register bits PR00 and PR01).

PRSL bit = 0:

Multiple interrupt servicing control is not performed for level 3 maskable interrupts.

In this mode, if the interrupt enabled state is in effect (IE = 1) while a previous level 3 maskable interrupt is being serviced, an interrupt request of the same level (level 3) can be acknowledged. (If the interrupt currently being processed is generated again, the same interrupt is acknowledged again.)

However, while a maskable interrupt of level 2 or above is being serviced, a level 3 interrupt request is held pending. In this mode, higher-level interrupt requests can be acknowledged. If a level 2 or higher maskable interrupt is being serviced, interrupts of the same level are disabled and interrupt requests are held pending.

In this mode, the value of bit 3 of the in-service priority register (ISPR) is undefined.

See **5.5 Control Registers** for details of the interrupt mode control register (IMC).

5.4 Interrupt Response Methods

The V55PI uses three different interrupt response methods: a vectored interrupt function, register bank switching function, and macro service function. The appropriate method can be selected for each interrupt by means of the interrupt request control register (IC××) to suit the purpose for which a particular interrupt is used. The interrupt controller handles interrupt requests in accordance with the set response method.

A comparison between the V55PI and V25/V35 interrupt response methods is shown below.

(1) Vectored interrupt functions

- Same functions as V25/V35
- Acknowledged in EI state only
- Interrupt requests held pending in DI state
- Priority: Specified by default priority and priority specification flag
- Nesting in interrupt specified as higher priority

(2) Register bank switching function

- Increased number of register banks (V25: 8 banks → V55PI: 16 banks)
- Different register bank specification method from V25/V35
- The register bank switched to by an interrupt is set in the vector table (memory) offset address (low-order bits 0 to 3) for each interrupt request
  (In the V25/V35: Individual input priority level = register bank)
- Other functions are the same as for the vectored interrupt function

(3) Macro service function

- Macro service types greatly increased compared with V25/V35
- The macro service control word is mapped onto the register file space (internal RAM) along with functional extension (V25/V35: Mapped onto the special function register area)
- Acknowledgment possible in either EI or DI state (acknowledged also possible from NMI or INTWDT service routine)
- Priority: Default priority only valid
- No nesting (as processing is performed by microprogram, not an instruction)

5.4.1 Vectored interrupts

A vectored interrupt can only be acknowledged in the interrupt enabled state (EI state). When a vectored interrupt request is acknowledged, the interrupt disabled state (DI state) is set, and the current PSW contents and PC and PS contents are saved to the stack. Then the corresponding vector is selected from the vector table, and the interrupt service routine is started at the address indicated by that vector. Vector numbers are fixed for each interrupt source. In the DI state, an interrupt request is held pending, and acknowledged when the EI state is set.

The return from the interrupt is performed by a RETI instruction. When returning from a hardware interrupt other than a non-maskable interrupt, an FINT instruction must be executed before executing an RETI instruction. When a return is made from an interrupt, the PC, PS and PSW are restored from the stack.

Figure 5-2. Interrupt Acknowledge Operation (Performed in Sequence ① → ④)

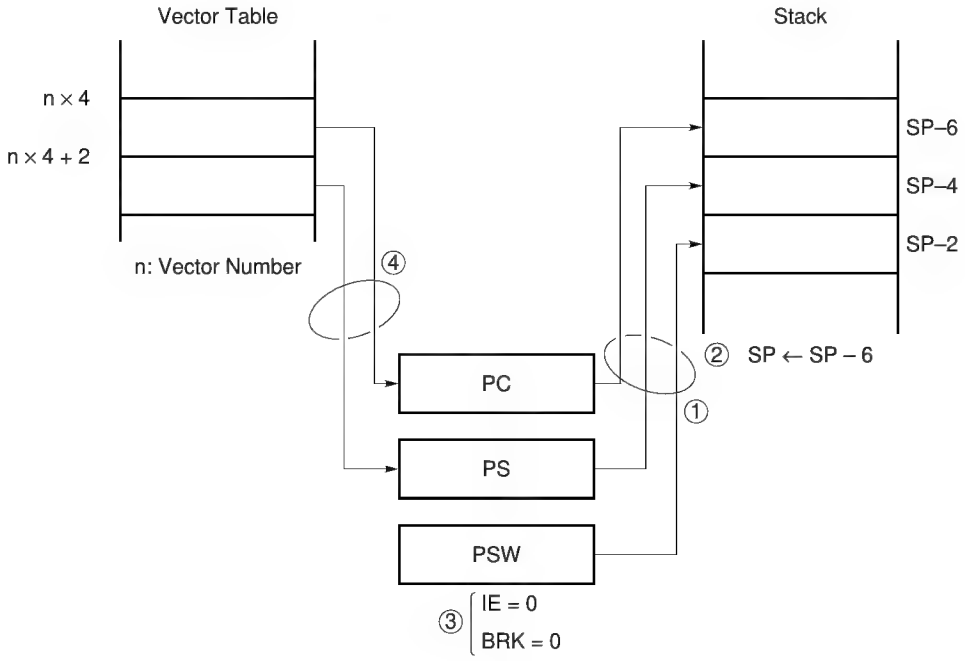
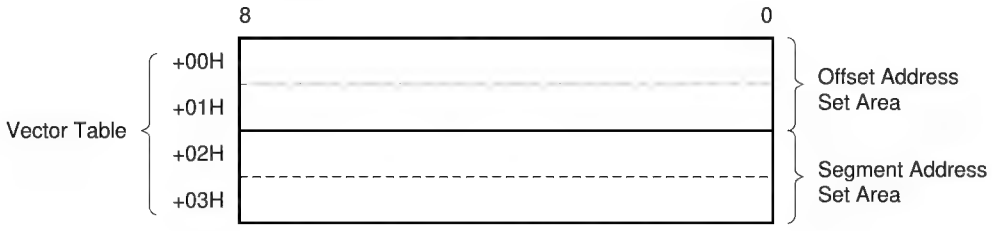


Figure 5-3. Vector Table Configurations



5.4.2 Register bank switching

In the V55PI, general register sets are mapped onto internal RAM, and register sets can be held in up to 16 banks. Interrupt servicing is performed by automatically switching the register bank when an interrupt is responded to. Because saving of registers to the stack previously performed by software is not required, high-speed switching of the program environment is possible.

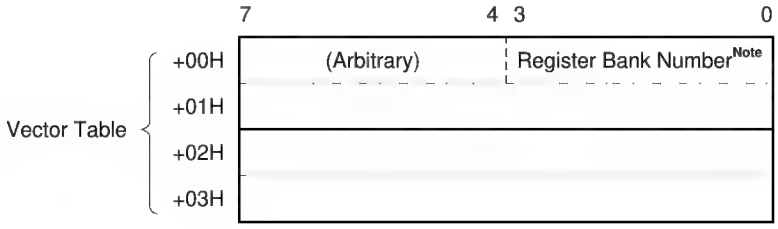
The register bank switching can be performed by BRKCS and TSKSW instruction executions.

As the register bank shares the DS3 area with the vector PC, when using the register bank for interrupt and in addition using the DS3 during the interrupt, the vector PC must be saved in the stack.

(1) Register bank switching by interrupt

When the register bank switching function is used when responding to an interrupt, bit 4 (ENCS) of the interrupt request control register corresponding to the interrupt request is set to "1", and the start of register bank switching is enabled. The number of the register bank to be switched to by the interrupt is set in the low-order 4 bits of the start address of the vector table for each interrupt source. When a specified interrupt request is generated in the interrupt enabled state (IE = 1) and in the interrupt mask release state, the register bank is switched.

Figure 5-4. Vector Table and Register Bank Number



Note As the low-order 4 bits of the interrupt vector address are used for the number of the register bank to be switched to, the following should be observed when both vectored interruption and register bank switching are used for the same interrupt request.
(Low-order 4 bits of interrupt service routine start address) = (register bank number to be switched to)

The register bank switching sequence is as follows (see Figure 5-5):

<1> PSW contents are saved to a temporary register.
<2> The register bank is switched.
<3> IE and BRK are set to 0.
<4> The PC contents and the PSW contents saved in the temporary register are saved to the respective save areas in the register bank.
<5> The offset of the start address of the interrupt service routine is loaded into the PC from the vector PC area in the register bank.

The above sequence switches the register bank and starts the interrupt service routine execution.

As can be seen from this sequence, when the register bank switching function is used, the PS, vector PC, SS, and SP in the register bank to be switched to need to be initialized beforehand. However, when the MOVSPA instruction is used at the start of the interrupt service routine, it is not necessary to initialize SS and SP. Other registers should be initialized as required. Note that the PS should not be changed within an interrupt service routine. If the PS is changed and the interrupt is ended in that state, when that interrupt is acknowledged at the next time, the PS value is the changed value and the operation is not transferred to the desired address.

Because an interrupt which is the source of register bank switching is subject to multiple interrupt service control, the return from register bank switching due to an interrupt is performed by executing an FINT instruction, then an RETRBI instruction. At this time, the FINT instruction should be executed immediately before the RETRBI instruction.

When the RETRBI instruction is executed, the PC and PSW are restored from the respective save areas in the register bank, as shown in Figure 5-6. Because the RETI instruction does not perform register bank restoration, a normal return to the main routine is not possible.

Figure 5-5. Register Bank Switching Sequence (Register Bank Switching by Interrupt)

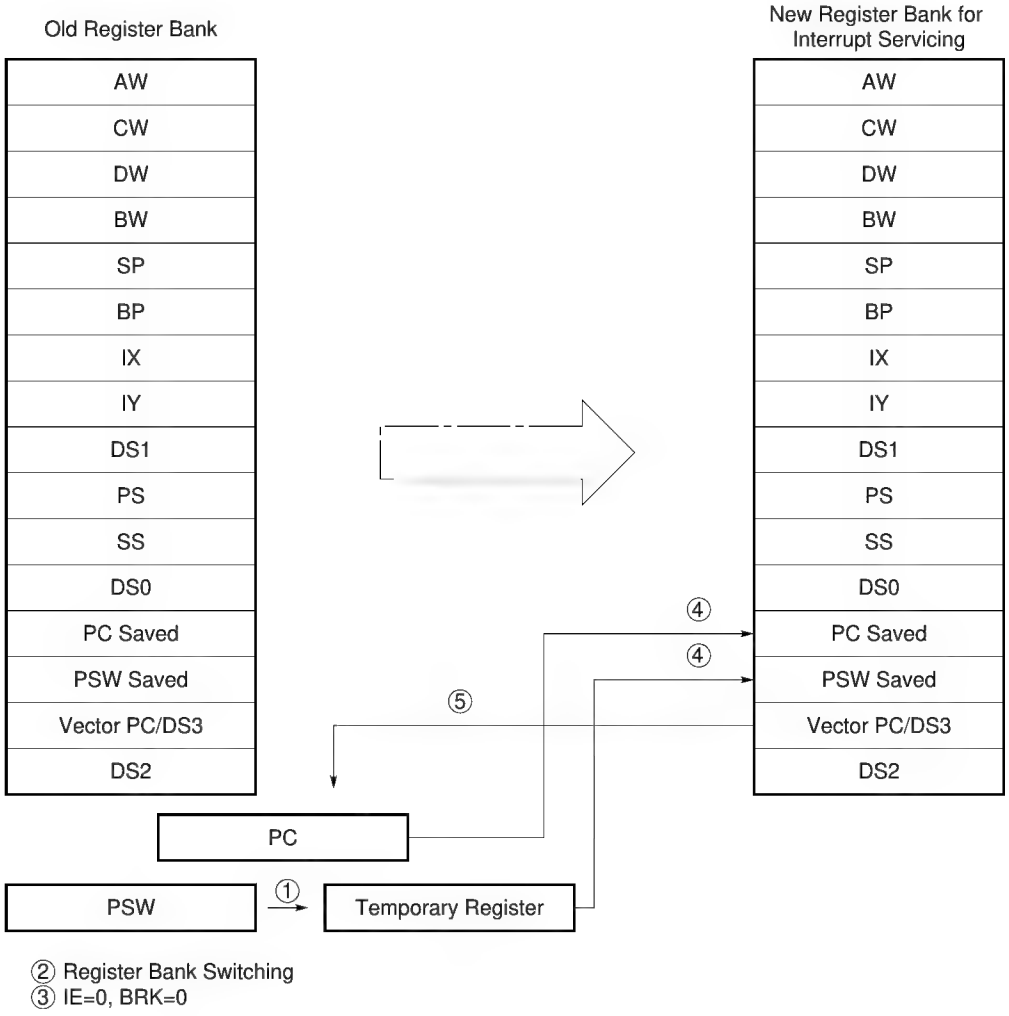
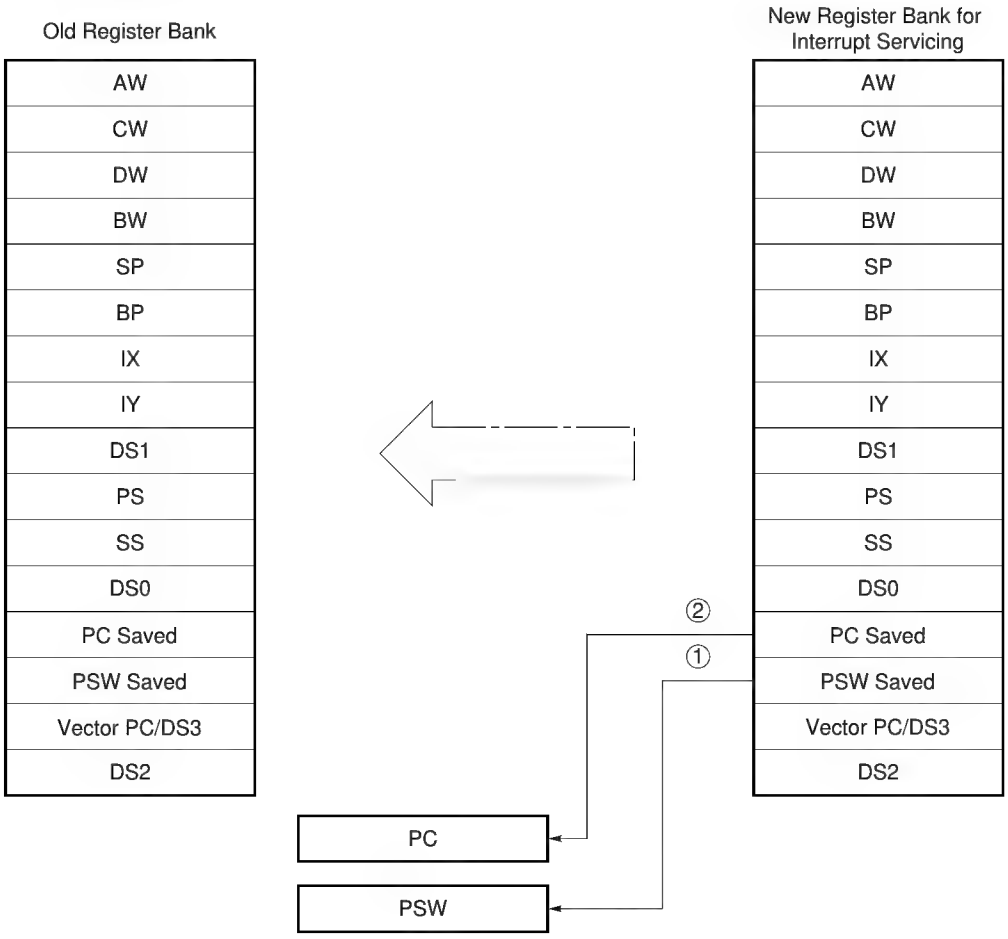


Figure 5-6. Register Bank Restoration Sequence

Execution of the RETRBI instruction performs register restoration processing in the sequence shown below.



(2) Register bank switching by software

(a) Switching by BRKCS instruction (BRKCS reg16)

Executing the BRKCS instruction enables register bank switching to be performed in the same sequence as in the case of register bank switching due to an interrupt. Therefore, it can be used for high-speed subroutine call.

The register bank is selected by means of the low-order 4 bits of the 16-bit register specified in the operand. When the BRKCS instruction is executed, the PS and vector PC in the register bank to be selected must be initialized beforehand. The SS and SP should be initialized either before or after switching. When an MOVSPA/MOVSPB instruction is executed in initialization, the values prior to switching can be set to both SS and SP of the switching destination, and therefore the stack can be used sequentially before and after switching.

The return from the new register bank is performed by execution of an RETRBI instruction. An FINT instruction need not to be executed at this time.

(b) Switching by TSKSW instruction (TSKSW reg16)

Register bank switching can be performed by executing the TSKSW instruction. When performing a multi-task control, the program execution information (context) of each task is assigned to the corresponding register bank, allowing the task to be switched at high speed.

The register bank is selected by means of the low-order 4 bits of the 16-bit register specified in the operand.

The register bank switching sequence is as follows:

<1> The PC and PSW are saved in the PC save area and PSW save area in the current (pre-switching) register bank.

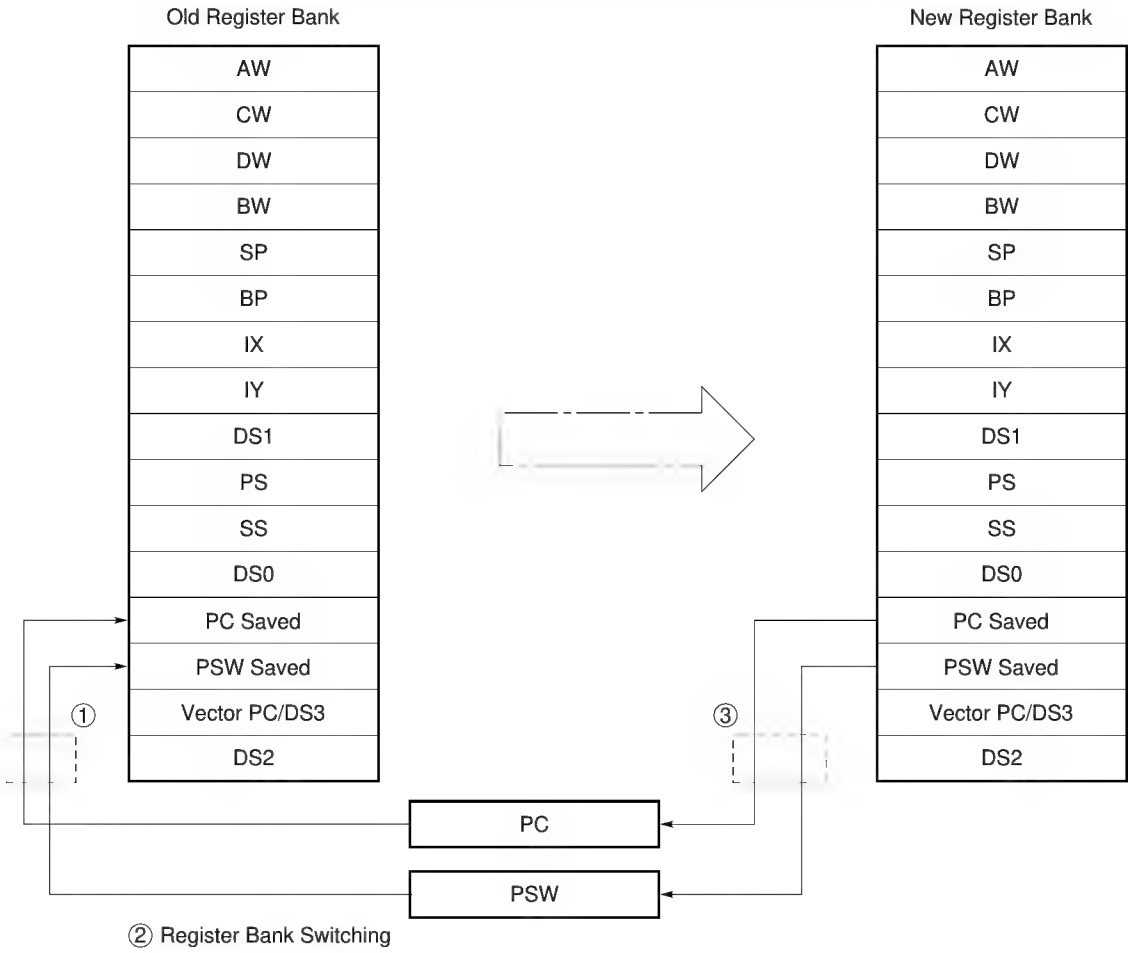
<2> The register bank is switched.

<3> The value of the PSW save area in the new register bank is loaded into the PSW, and the value of the PC save area is loaded into the PC and a branch is performed.

When the TSKSW instruction is executed, the PS and PC save areas, SS, and SP in the register bank to be selected before the first switchover must be initialized beforehand. If the SS and SP are initialized by execution of a MOVSPB instruction, the pre-switching values can be set in the switching destination SS and SP.

The register bank switching sequence is shown in the following Figure 5-7.

Figure 5-7. Sequence for Register Bank Switching by TSKSW Instruction

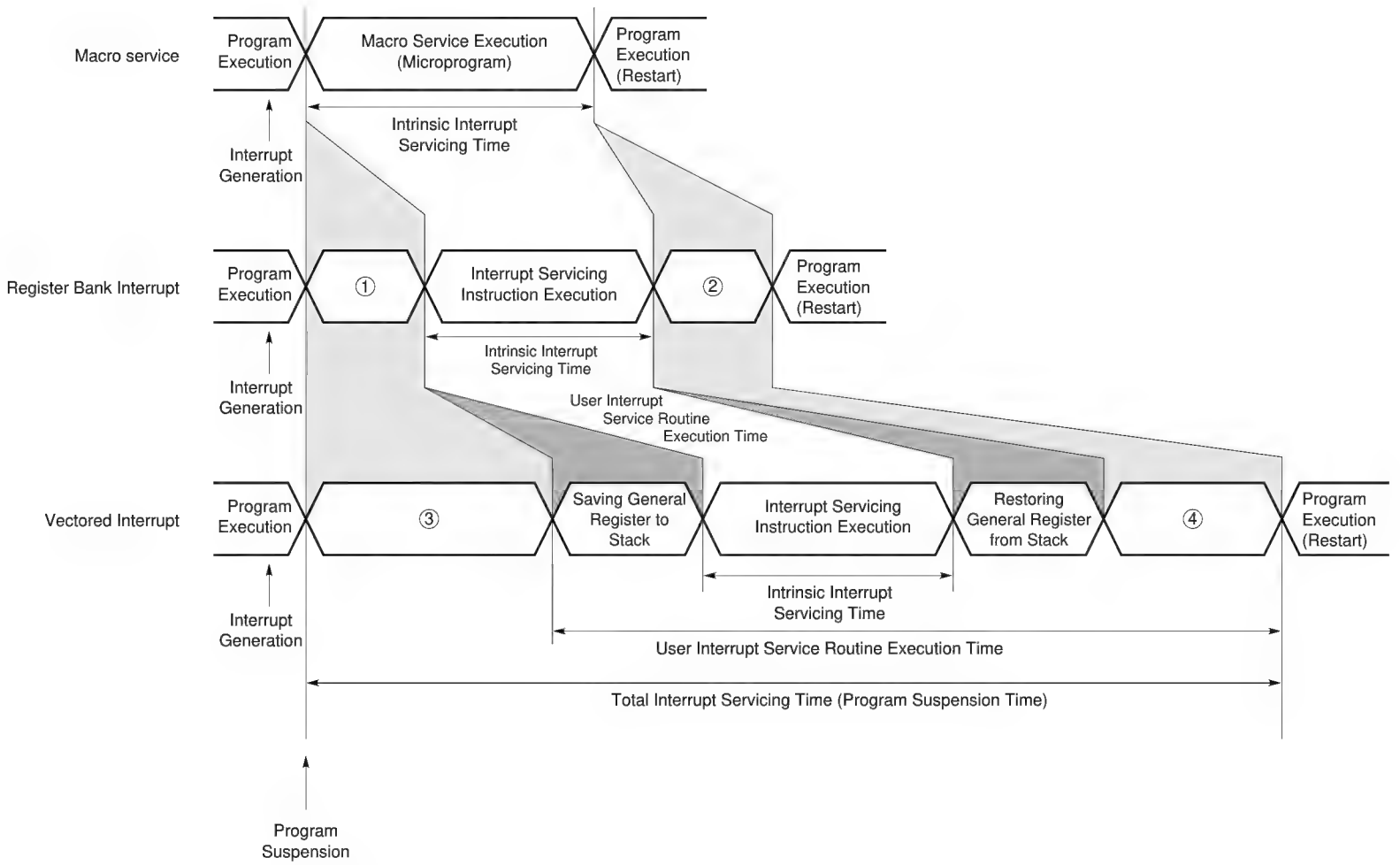


5.4.3 Macro service function

The macro service function performs processing of simple data transfers, etc., by means of a microprogram (CPU internal dedicated firmware) started by generation of an interrupt request. By means of this function, simple and stereotyped interrupt servicing which was previously coded and executed by a user program, is automatically processed.

Macro service processing is processed in response to an interrupt request. This function is intended to minimize the frequency of branching to the interrupt service routine of a user program which is centered on software processing and to suppress the software overhead caused by a series of processes such as register save, initialization, register restoration, and return from an interrupt routine, for improving the CPU efficiency (see **Figure 5-8**).

Figure 5-8. Interrupt Servicing Efficiency with Macro Service



① Saving PC, PSW to register file. Reading and branching register number from vector table.
② RETRBI instruction execution. Restoring and branching PC, PSW from register file.
③ Saving PC, PS, PSW to stack. Reading and branching PC, PS from vector table.
④ RETI instruction execution. Restoring and branching PC, PS, PSW from stack.

Processing performed by the macro service is transparent in terms of software, and it is possible to process as a single mass of data what was previously processed by software in byte units, allowing more efficient programming.

The V55PI macro service supports not only the simple data transfers used in the V25 and V35, but also the data compare, calculating the data difference, and various operating modes closely linked to the V55PI on-chip peripheral hardware.

(a) EVTCNT (EVENT COUNTER)

The counter is updated each time of macro services are generated, and when the counter reaches 0 macro service for the corresponding interrupt source is ended and a vectored interrupt or register bank switching is generated.

(b) DTACMP (DATA COMPARE)

The interrupt source specific SFR and preset byte data are compared, and if they match, macro service for the corresponding interrupt source is ended and a vectored interrupt or register bank switching is generated.

(c) DTADIF (DATA DIFFERENCE)

Calculates the difference using the capture register of the timer/counter unit. DTADIF is started by a timer interrupt and subtracts the value of the previously latched capture register from the value of the capture register latched this time and stores the result in the predetermined memory buffer.

After completing processing the number of predetermined times, DTADIF terminates macro service of the corresponding interrupt source and generates vectored interrupts or register bank switching.

(d) BLKTRS (BLOCK TRANSFER)

Data transfer is performed between the preset SFR and the preset memory buffer.

When the preset number of data transfers have been performed, macro service for the corresponding interrupt source is ended and a vectored interrupt or register bank switching is generated.

(e) BLKTRS-C (BLOCK TRANSFER WITH CHARACTER SEARCH)

Performs a data transfer between the predetermined memory buffer and SFR. When the data transfer is completed by the number of predetermined times or when the transfer data matches the predetermined character data, BLKTRS-C terminates macro service of the corresponding interrupt source and generates a vectored interrupt or register bank switching.

(f) RTOPTRN (RTOP TRANSFER)

Data to be output to the real-time output port is transferred to the port 7 buffer (P7H, P7L), and the interval specification data for output to the real-time output port is transferred to the timer compare register (CM00, CM01).

(g) DTACMP-M (DATA COMPARE WITH CHARACTER MASK)

The logical product (AND) of the status data read from the external I/O and the preset mask data is obtained. The result is compared with the preset byte data. If they match, data is transferred between the external I/O and memory. If they do not match, or the preset number of data transfers has been performed, the macro service of the corresponding interrupt source is ended and a vectored interrupt or register bank switching is generated.

(1) Macro service processing

The macro service function is controlled by macro service control registers provided for each interrupt source for which macro service can be used.

Macro service processing is started when the MS/$\overline{\text{INT}}$ bit in the interrupt request control register (IC××) is set to the macro service specification status (= 1). When an interrupt request is generated, the processing specified by the macro service mode register in internal RAM (register file) is started with the program status information (PC, PSW) retained unchanged, and on completion of this processing an instruction execution or other interrupt servicing is performed. Macro service is executed regardless of whether the interrupt enabled state (IE = 1) or interrupt disabled state (IE = 0) is in effect.

(2) Macro service completion interrupt

When the macro service counter (MSC) = 0 or conditions for ending the macro service is established, the MS/$\overline{\text{INT}}$ bit is cleared with the corresponding interrupt request control register IF bit set, and a vectored interrupt or register bank switching (macro service completion interrupt) is generated.

When an interrupt request is acknowledged, the IF bit is cleared in the same way as the normal interrupt. To perform macro service in succession, set the parameters required in this interrupt service routine again and set the MS/$\overline{\text{INT}}$ bit.

The macro service completion interrupt is controlled by the priority and interrupt disabled (DI) states. Therefore, if the same interrupt is generated while macro service completion interrupt is pendent the second interrupt generated is ignored. (Excluding UART transmission interrupt and UART reception interrupt. Refer to **7.4.8 (6) Transmit/receive interrupt**.)

If the same interrupt request is generated during the macro service completion interrupt servicing, it is acknowledged as follows.

- When the MS/$\overline{\text{INT}}$ bit is set after the interrupt request generation
  IF bit = 1 and the interrupt request is pendent. Macro service is executed when MS/$\overline{\text{INT}}$ bit is set.
- When the interrupt request is generated after MS/$\overline{\text{INT}}$ bit is set
  Macro service is executed when the interrupt request is generated.
- When the interrupt operation is ended while MS/$\overline{\text{INT}}$ bit = 0
  IF bit = 1 and the interrupt is pendent until the executed interrupt operation ends.
  When the interrupt operation ends, the pending interrupt is acknowledged.

(3) Macro service control registers

A macro service control word specified for each interrupt source and a macro service channel specified for each operating mode are used for macro service operation specification and other control.

(a) Macro service control word

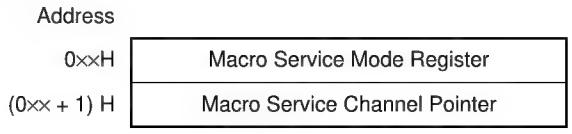
The macro service control word is located in the lower 64 bytes (000H to 03FH) of the internal RAM area (register file space) overlapped with the register banks 0 and 1.

Of the V55PI's 25 maskable interrupt requests, macro service is supported for the 22 sources shown in Table 5-2. The macro service control word address used for each interrupt request is fixed.

The macro service control word is composed of the macro service mode register which controls macro service functions and the macro service channel pointer.

The basic configuration of the macro service control word is shown in Figure 5-9.

Figure 5-9. Basic Configuration of Macro Service Control Word



The macro service mode register sets the macro service mode, and the macro service channel pointer specifies the macro service channel address to be used for each operating mode.

When macro service processing is performed, the macro service mode register and channel pointer must be initialized beforehand for interrupt requests for which macro service processing can be specified.

If there are interrupt sources for which the macro service function is not used, the macro service control words can be used freely as ordinary internal RAM.

Table 5-2. Interrupt Requests and Macro Service Control Word Addresses

Control Word Address	Interrupt Request	Interrupt Generation Source
008H	INTPAI	Parallel interface
00AH	INTAD	A/D converter
012H	INTP0	INTP0 pin input
014H	INTP1	INTP1 pin input
016H	INTP2	INTP2 pin input
018H	INTP3	INTP3 pin input
01AH	INTP4	INTP4 pin input
01CH	INTP5	INTP5 pin input
020H	INTCM00	CM00 match detection
022H	INTCM01	CM01 match detection
024H	INTCM10	CM10 match detection
026H	INTCM11	CM11 match detection
028H	INTCM21	CM21 match detection
02AH	INTCM31	CM31 match detection
02CH	INTD0	DMA channel 0 main
02EH	INTD0S	DMA channel 0 sub
030H	INTD1	DMA channel 1 main
032H	INTD1S	DMA channel 1 sub
038H	INTSR0/ INTCSI0	UART channel 0 reception/ serial channel 0 transmission/reception
03AH	INTSR1/ INTCSI1	UART channel 1 reception/ serial channel 1 transmission/reception
03CH	INTST0	UART channel 0 transmission
03EH	INTST1	UART channel 1 transmission

Caution Addresses 000H to 007H are used as a work area when macro service (RTOPTRN) is executed.

(b) Macro service mode register

The macro service mode register is an 8-bit register which specifies the macro service operation. It is mapped onto internal RAM as a part of the macro service control word (see Figure 5-9).

(c) Macro service channel pointer

The macro service channel pointer specifies the start address of the macro service channel in the lower 256 bytes of the register file space.

This pointer is mapped onto internal RAM as part of the macro service control word.

(d) Macro service channel

The macro service channel is a structure comprising the macro service counter, buffer, etc., and has a different structure in each macro service mode. The macro service channel is located in the lower 256 bytes (000H to 0FFH) of the register file space. The macro service channel must be specified so as to start at an even address.

(4) Macro service operating modes

The macro service operation is specified by the macro service mode register.

Table 5-3. Macro Service Operating Modes

Value Set in Macro Service Mode Register	Function	Channel Specification
0000 00C0	EVTCNT (EVENT COUNTER)	No
0000 1000	DTACMP (DATA COMPARE)	Yes
0001 00W0	DTADIF (DATA DIFFERENCE)	
010M D0W0	BLKTRS (BLOCK TRANSFER)	
011M D0W0	BLKTRS-C (BLKTRS WITH CHARACTER SEARCH)	
1000 0UA0	RTOPTRN (REAL-TIME OUTPUT PORT TRANSFER)	
001M 00d0	DTACMP-M (DATA COMPARE WITH CHARACTER MASK)	

Caution Note that if a code other than those shown above is specified, the macro service function and subsequent operations cannot be guaranteed.

For macro service operating mode set values, the operation is determined by specifying the following subcommands.

C: Macro service counter (MSC) increment (+1)/decrement (–1) specification

C	Operation
0	Increment
1	Decrement

W: Data byte/word specification

W	Byte/Word
0	Byte
1	Word

D, d: Data transfer direction specification

D	Transfer Direction
0	Memory ← SFR
1	SFR ← Memory

d	Transfer Direction
0	Memory ← I/O
1	I/O ← Memory

M: Memory space address mode specification

M	Address Mode
0	1-Mbyte basic memory space access mode
1	16-Mbyte expansion memory space access mode

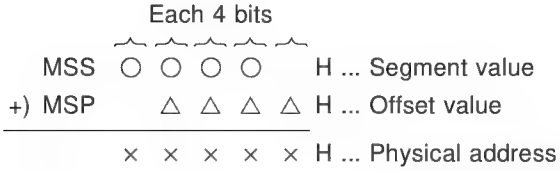
U: Compare register set data table address update setting

U	Compare Register Set Data Reference Address
0	MEM.PTR (segment: MTS, offset: MTP) not updated
1	MEM.PTR – MSC × 2 + 2 (segment: MTS, offset: MTP) address updating

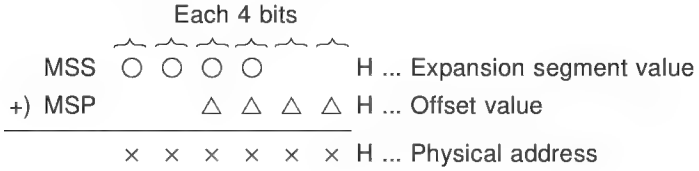
A: Compare register set data manipulation specification

A	Compare Register Set Data
0	Set data in table transferred to compare register as it is
1	Set data in table + current compare register set data transferred

1-Mbyte Basic Memory Space Access Mode



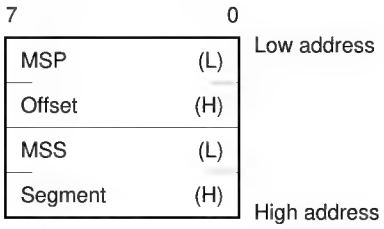
16-Mbyte Expansion Memory Space Access Mode



(5) Addressing methods

Data addressing methods with the macro service channel are shown below.

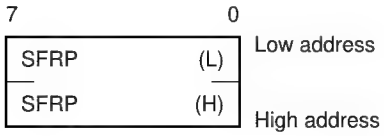
(a) External memory access (= MEM1.PTR)



An SFR can also be made the object of an access.

(b) SFR access (= SFRP)

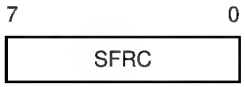
The SFR address specification pointer is composed of 2 bytes. Set the low-order 16 bits of the address of the special function register (SFR) to which data is transferred. Set 1 to all of the high-order 7 bits.



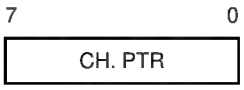
Example To specify RxB0 (UART receive buffer: 0FFF76H)
SFRP(L) 76H
SFRP(H) 0FFH
To specify PAB (Parallel interface buffer: 0FFE10H)
SFRP FE10H

(c) Timer compare register access (= SFRC)

- Used in RTOPTRN mode
- Only the lower 8 bits can be specified and CM0 or CM1 selected for the compare register address.



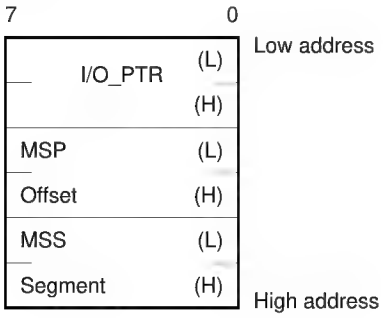
(d) Internal RAM access



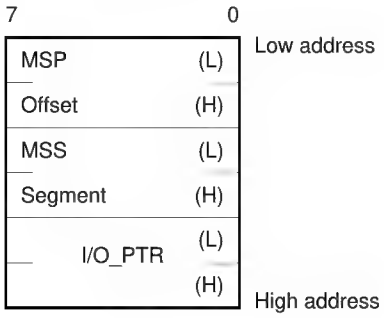
The area subject to internal RAM addressing is limited as the lower 256 bytes (000H to 0FFH).

(e) External I/O access (= I/O_PTR)

When performing I/O → memory transfer



When performing memory → I/O transfer

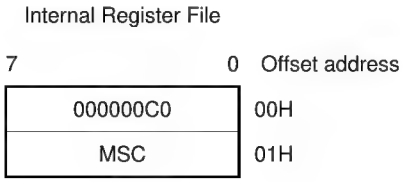


(6) Macro service operations

(a) EVTCNT (EVENT COUNTER)

After the preset number of macro services are generated, macro service for the corresponding interrupt source is terminated and a vectored interrupt or register bank switching is generated.

[Macro service control word]



[Operation]

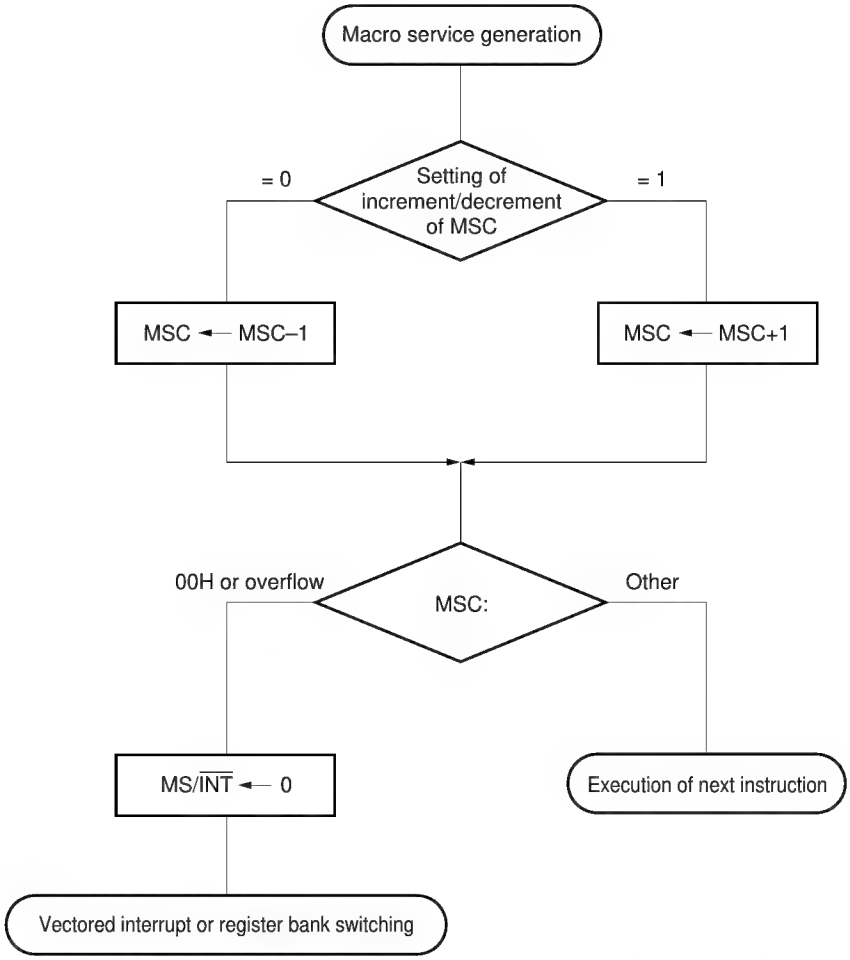
<1> The macro service counter (MSC) is incremented (+1) or decremented (–1) each time a macro service is generated.

<2> When the MSC reaches 00H or overflows, the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0 and a vectored interrupt request or register bank switching request is generated. When decrementing is performed, if 0 is set as the initial value of the MSC, processing is performed 256 times.

Whether the MSC is incremented or decremented depends on the macro service mode register subcommand C specification.

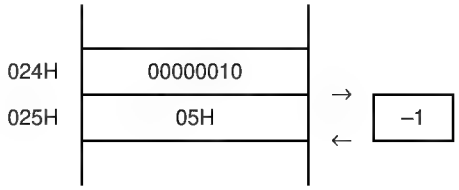
Table 5-4. EVENT Mode Operation Specification

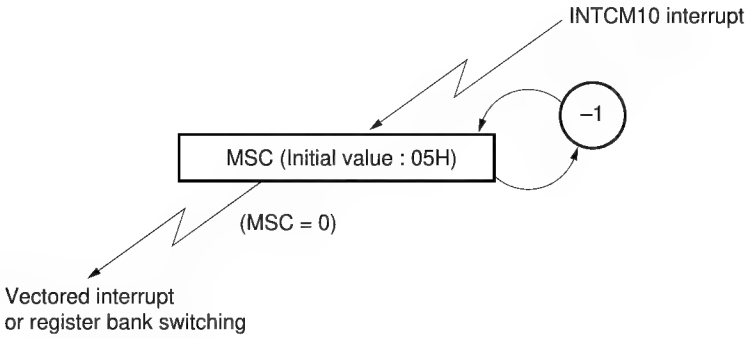
C	Operation
0	Increment
1	Decrement



In this mode, the macro service functions as a counter which divides the number of interrupt generations.

Example To perform division-by-5 of the number of INTCM10 interrupt requests generated.





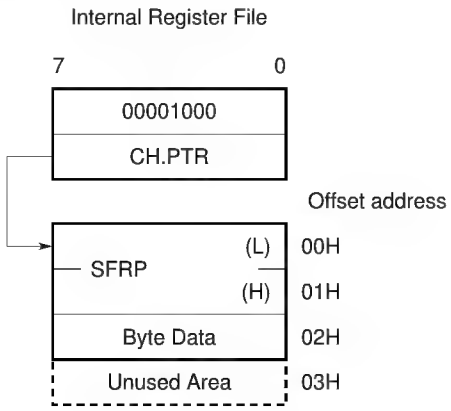
[Use]

Event counter, measurement of number of captures, etc.

(b) DTACMP (DATA COMPARE)

The interrupt source specific SFR and preset byte data are compared, and if they match, macro service for the corresponding interrupt source is terminated and a vectored interrupt or register bank switching is generated.

[Macro service control word]

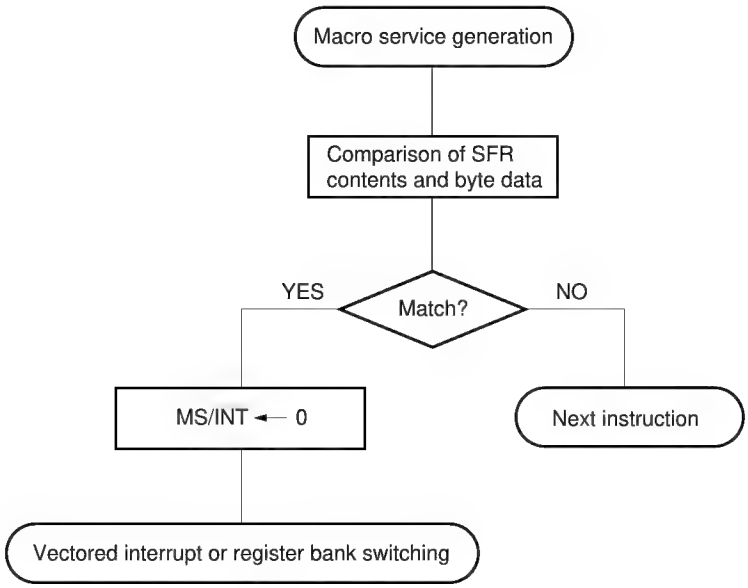


[Operation]

<1> The start address of the macro service channel is specified by the channel pointer (CH.PTR).

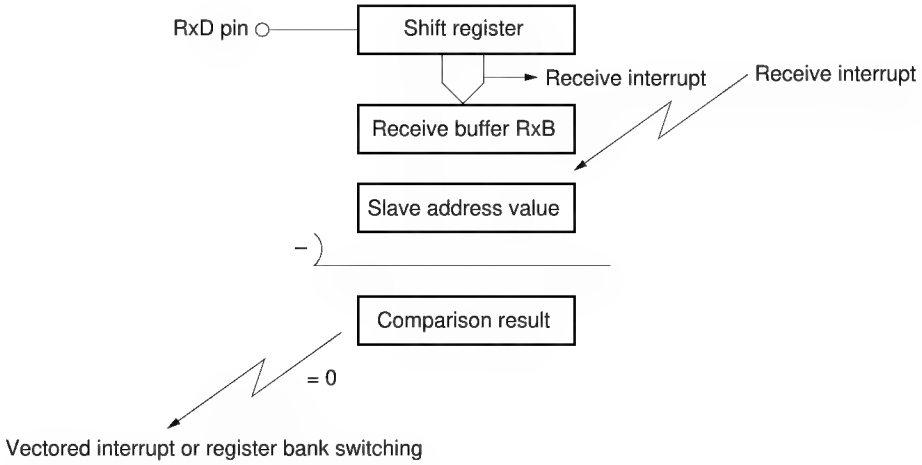
<2> When a macro service is generated, the contents of the SFR specified by the SFR pointer (SFRP) are compared with the byte data, and if they match the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0 and a vectored interrupt request or register bank switching request is generated. If they do not match, the macro service is exited with no further action taken, and processing is continued.

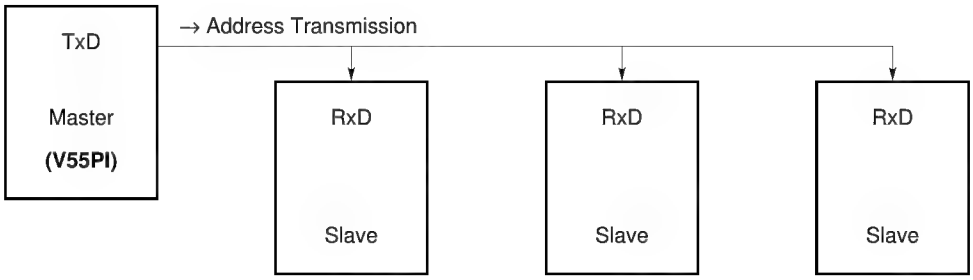
Caution Because the channel start address begins with an even byte, another macro service channel cannot be specified consecutively in the unused location.



Example Can be used for address match detection in a system which uses serial communication functions and performs address detection.

In the HALT mode, in particular, macro service is started by reception of a slave address, and if there is a match between the slave address in the receive buffer and the byte data, a vectored interrupt request or register bank switching request is generated. If there is no match, the HALT mode is entered again.





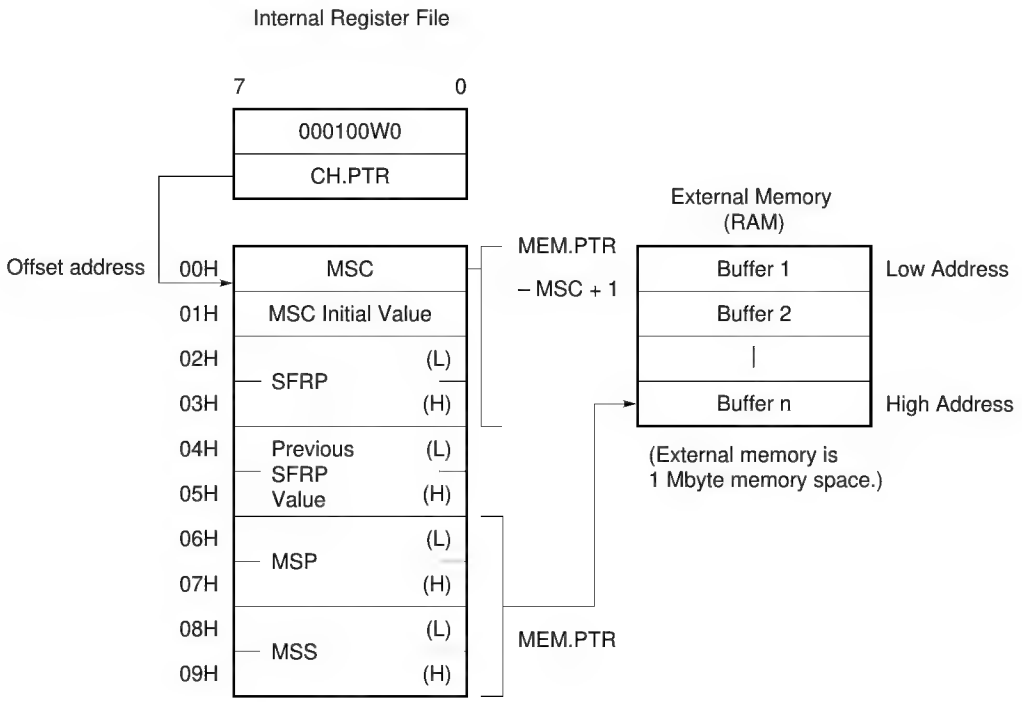
[Use]

Slave address detection and port input status polling, etc. in serial communications

(c) DTADIF (DATA DIFFERENCE)

Used for calculating the difference using the timer/counter unit capture register, etc.

[Macro service control word]



[Operation] [Operation]

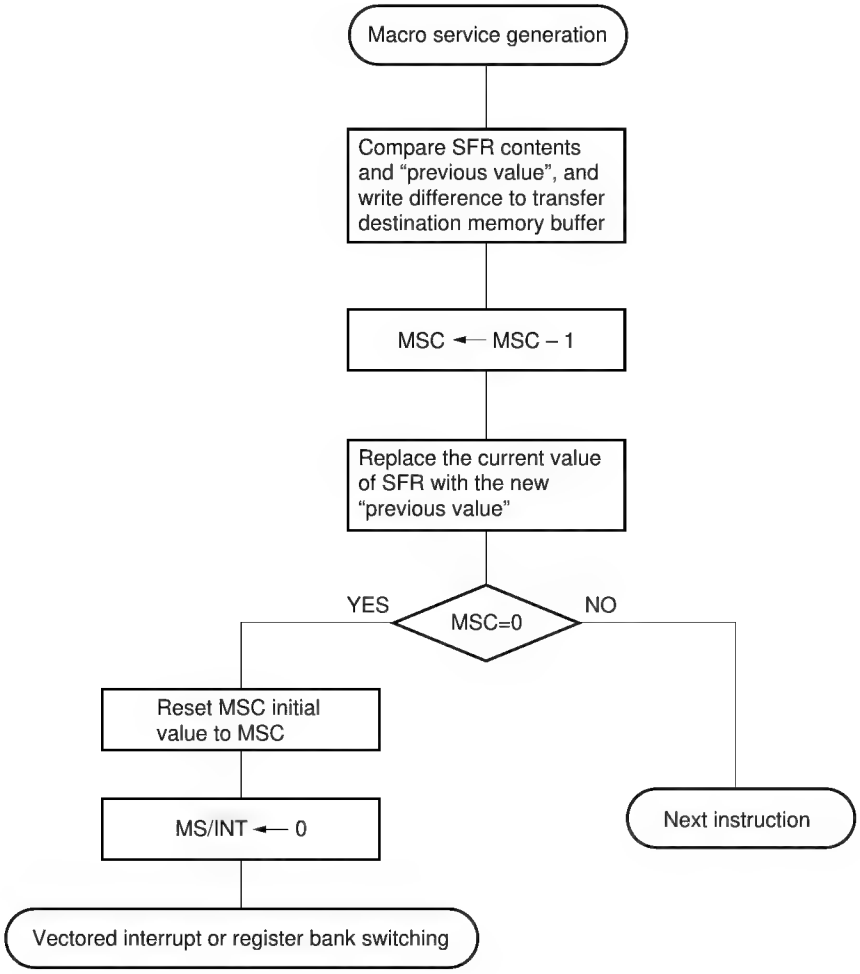
<1> The start address of the macro service channel is specified by the channel pointer (CH.PTR).

<2> The transfer destination external memory (buffer n) maximum address is specified by MEM.PTR [memory pointer (MSP) and memory segment (MSS)]. MEM.PTR can access the 1 Mbyte basic memory space.

<3> The difference between the current value of the SFR (e.g. capture register) specified by the SFRP and the "previous value" is written to the buffer memory. At this time, the data write starts with buffer 1. The MSC is decremented each time a data write is performed (byte/word), and subsequently the result is written to the buffer memory indicated by MEM.PTR – MSC + 1 in the case of a byte transfer, or MEM.PTR – MSC × 2 + 2 in the case of a word transfer.

<4> Next, the current SFR value is made new "previous value" and stored in the area. In byte mode, the "previous value" is stored in "previous SFRP Value (L)".

<5> When the MSC becomes 0, the initial MSC value is set in the MSC again, the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0, and a vectored interrupt request or register bank switching request is generated. If 0 is set as the initial value of the MSC, processing is performed 256 times.



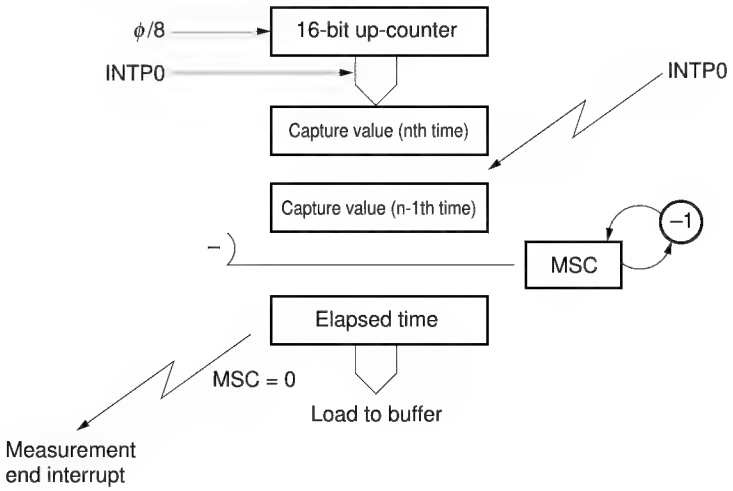
Caution With regard to external memory involved in a transfer, if the transfer address indicated by MSS and MSP hits a segment boundary, note that a transfer spanning segments cannot be performed.

Whether byte data or word data is handled depends on the subcommand W specification in the macro service mode register.

Table 5-5. Data Byte/Word Specification

W	Byte/Word
0	Byte
1	Word

Example Can be used for time measurement by INTP0.



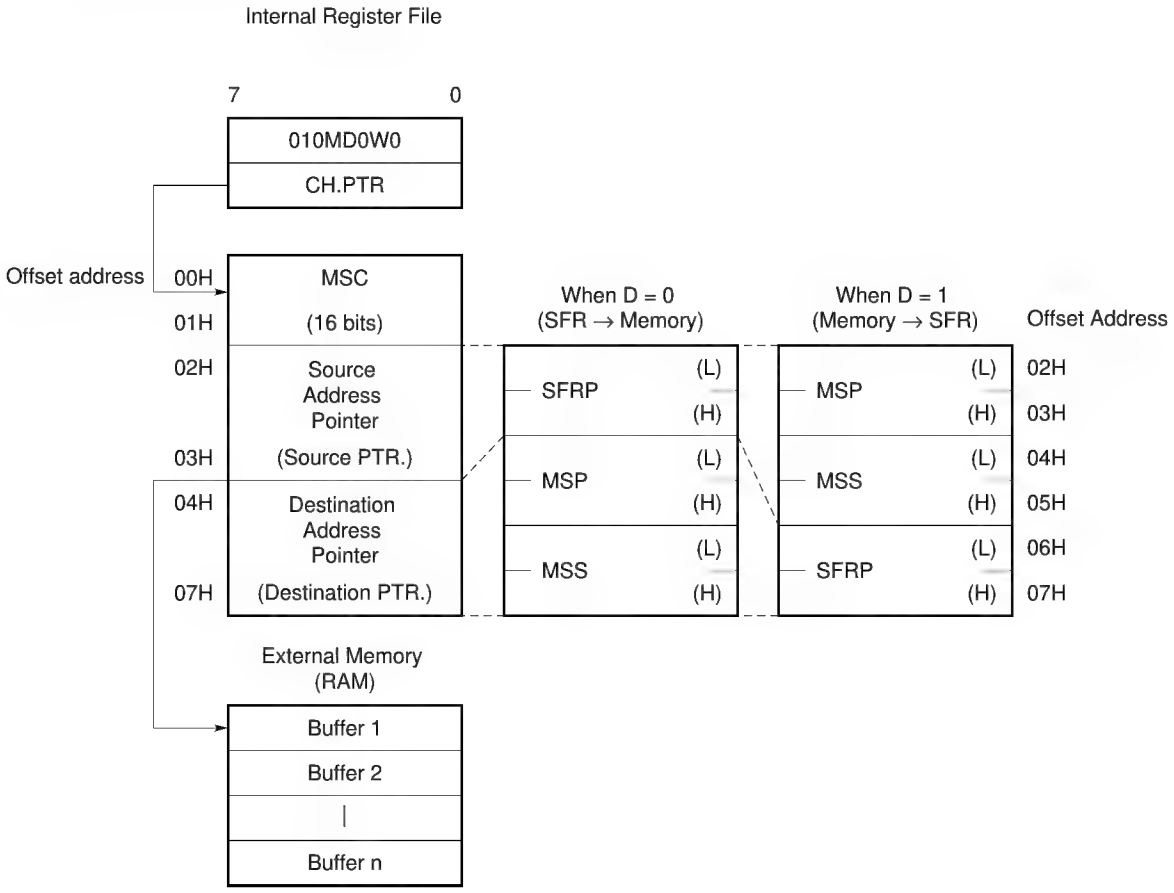
[Use]

Timing measurement using timer/counter unit capture function, measuring of change amount of conversion values by A/D converter, etc.

(d) BLKTRS (BLOCK TRANSFER)

Data transfer is performed between the pre-specified memory buffer and SFR. When the preset number of transfers have been performed, macro service for the corresponding interrupt source is terminated and a vectored interrupt or register bank switching is generated.

[Macro service control word]



[Operation] [Operation]

<1> The start address of the macro service channel is specified by the channel pointer (CH.PTR).

<2> Data transfer is performed between the SFR specified by the SFRP and the external memory buffer addressed by MEM.PTR [memory pointer (MSP) and memory segment (MSS)]. Data transfer starts from buffer 1.

<3> At the end of a byte/word data transfer, MEM.PTR is incremented (+1/+2).

<4> The macro service counter (MSC) is decremented (–1) each time data is transferred.

<5> When the MSC becomes 0, the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0 and a vectored interrupt request or register bank switching request is generated. If 0 is set as the initial value of the MSC, processing is performed 65536 times.

Caution With regard to external memory involved in a transfer, if the transfer address indicated by MSS and MSP hits a segment boundary, note that a transfer spanning segments cannot be performed.

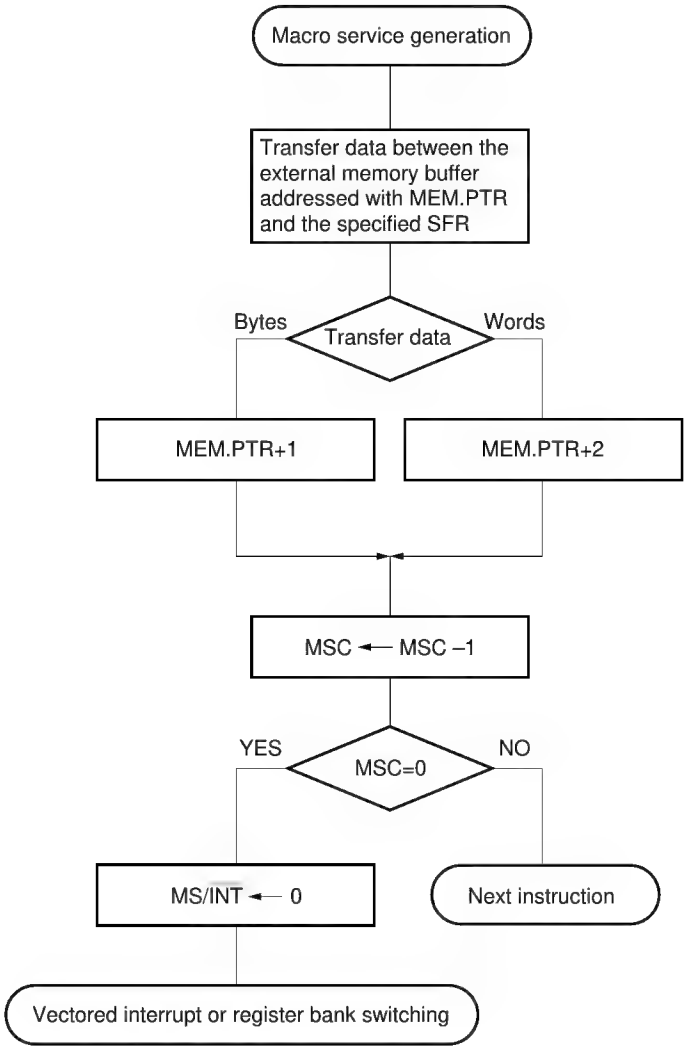


Table 5-6. BLKTRS Mode Operation Specification

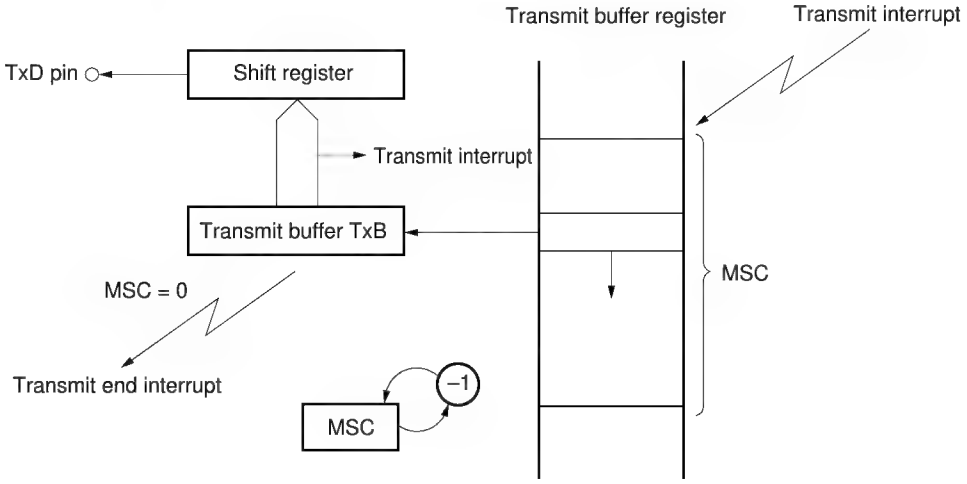
D	W	Operation	Transfer Data
0	0	Memory ← SFR	Byte
0	1	Memory ← SFR	Word
1	0	SFR ← Memory	Byte
1	1	SFR ← Memory	Word

Table 5-7. Memory Space Address Mode Specification

M	Address Mode
0	1-Mbyte basic memory space access mode
1	16-Mbyte expansion memory space access mode[Note]

Note The contents of the expansion segment registers (DS2, DS3) are irrelevant.

Example Can be used for UART data transmission.

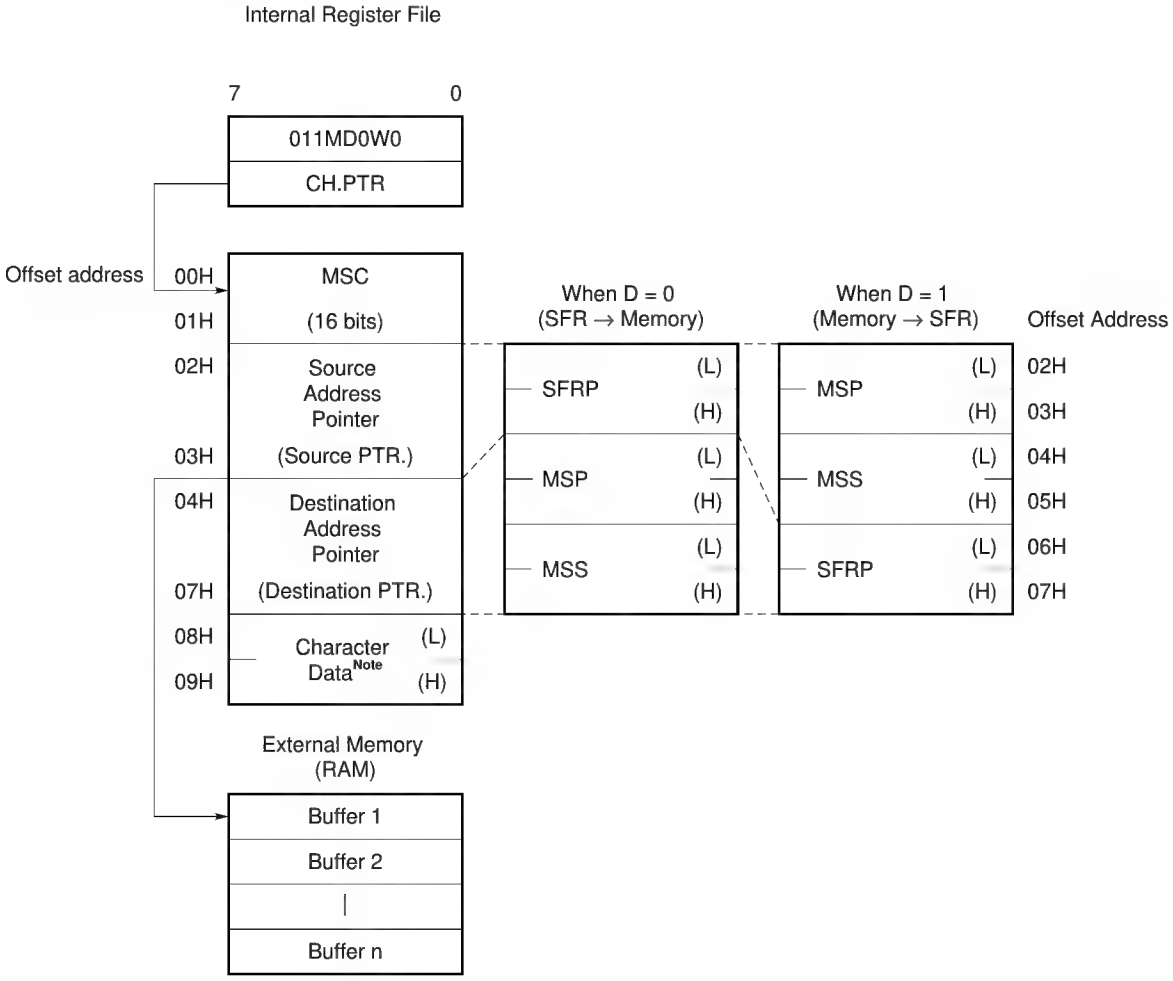


[Use]
UART data transmission/reception, etc.

(e) BLKTRS-C (BLKTRS WITH CHARACTER SEARCH)

Data transfer is performed between the pre-specified memory buffer and SFR. When the preset number of data transfers have been performed or the transfer data matches the preset character data, macro service for the corresponding interrupt source is terminated and a vectored interrupt or register bank switching is generated.

[Macro service control word]



Note This is specified as 1-byte data in the byte transfer mode.

[Operation]

<1> The start address of the macro service channel is specified by the channel pointer (CH.PTR).

<2> Data transfer is performed each time when macro service is generated between the SFR specified by the SFRP and the external memory buffer addressed by MEM.PTR [memory pointer (MSP) and memory segment (MSS)]. Data transfer starts from buffer 1.

<3> At the end of a byte/word data transfer, MEM.PTR is incremented (+1/+2).

<4> The macro service counter (MSC) is decremented (–1) each time data is transferred.

<5> Macro service processing continues until MSC = 0 or the same contents as the character data are transferred.

<6> When MSC = 0 or the same contents as the character data are transferred, the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0 and a vectored interrupt request or register bank switching request is generated. If 0 is set as the initial value of the MSC, processing is performed 65,536 times.

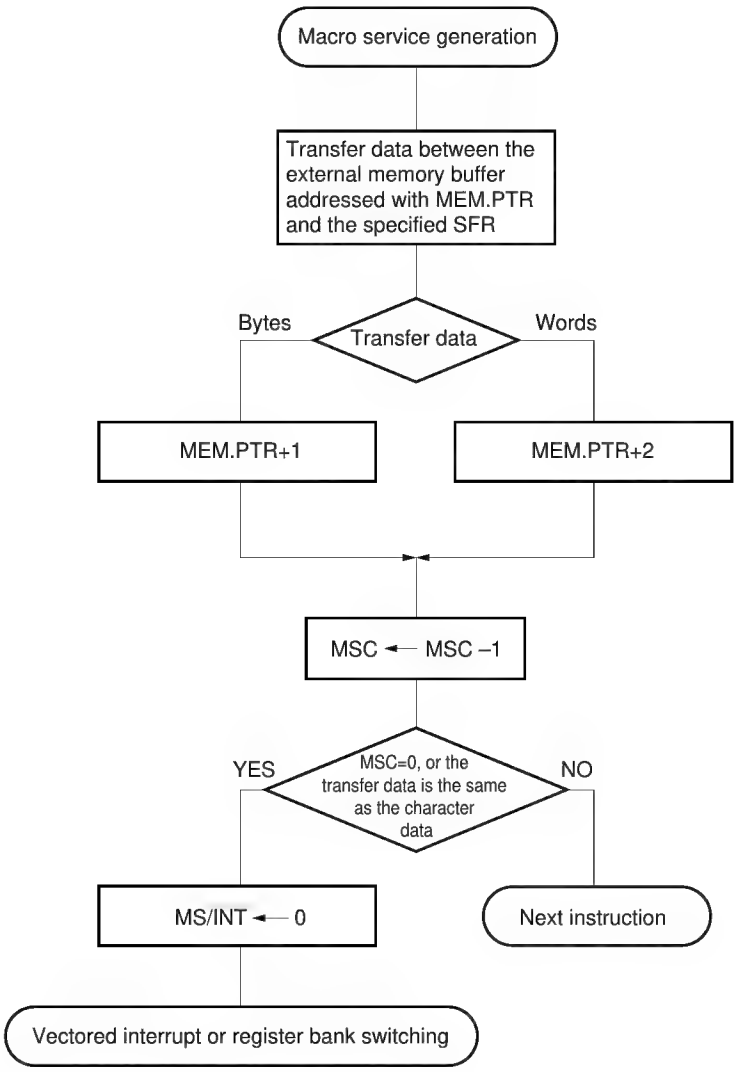


Table 5-8. BLKTRS-C Mode Operation Specification

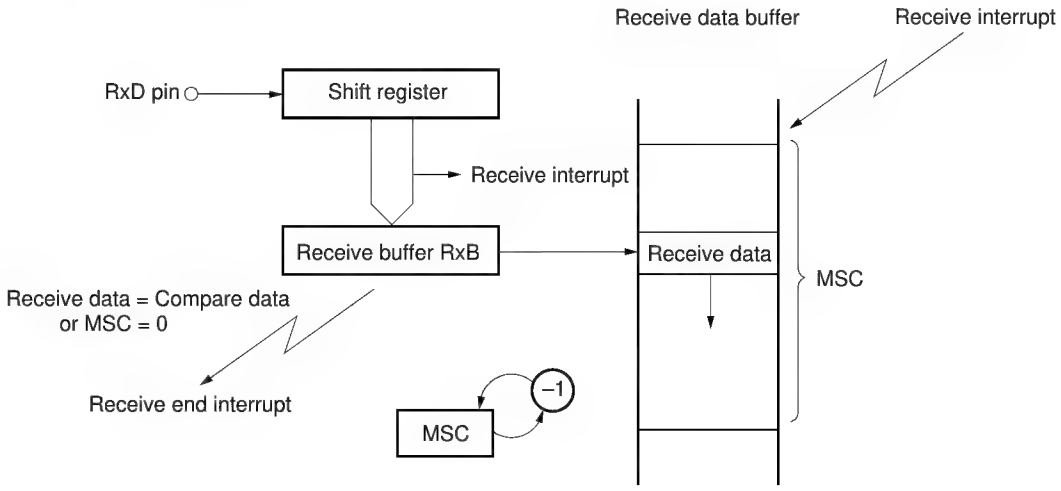
D	W	Operation	Transfer Data
0	0	Memory ← SFR	Byte
0	1	Memory ← SFR	Word
1	0	SFR ← Memory	Byte
1	1	SFR ← Memory	Word

Table 5-9. Memory Space Address Mode Specification

M	Address Mode
0	1-Mbyte main memory space access mode
1	16-Mbyte main memory space access mode[Note]

Note The contents of the expansion segment registers (DS2, DS3) are irrelevant.

Example Can be used for UART data reception (with completion code).



[Use]

UART data transmission/reception (with reception completion code detection function), etc.

(f) RTOPTRN (REAL TIME OUTPUT PORT TRANSFER)

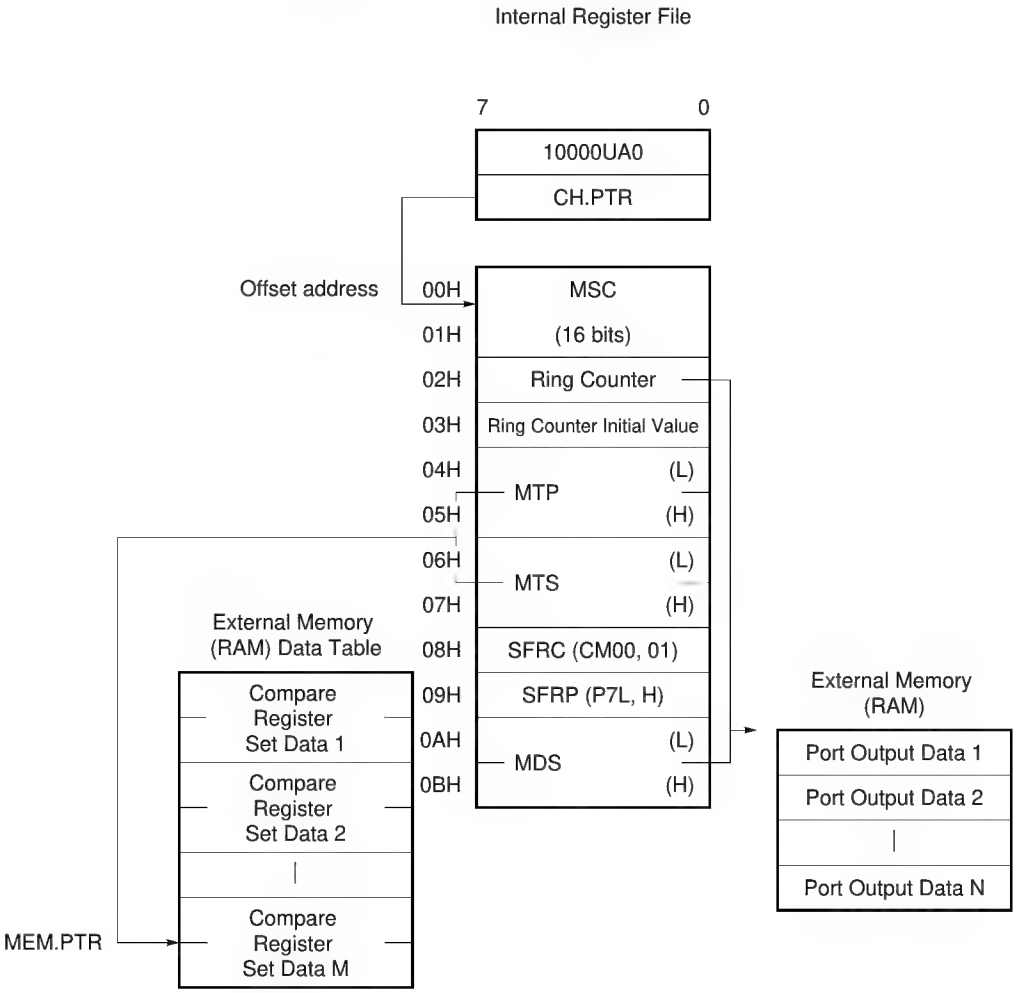
Transfer of data to be output to the real-time output port is performed.

The following two kinds of transfer are performed in response to one macro service request.

<1> The real-time output port output interval value is transferred from memory to an SFR (timer compare register).

<2> The real-time output port output data is transferred from memory to an SFR (P7L, P7H).

[Macro service control word]



Remark External memory is 1-Mbyte basic memory space.

[Operation]

The start address of the macro service channel is specified by the channel pointer (CH.PTR).

The following operations are performed in response to one macro service transfer request.

<1> Word data is transferred from the data table addressed by MTS and MTP to the timer compare register (CM00/CM01) specified by the SFRC.
The data table to be referenced and manipulation of data in the table can be specified by setting the macro service mode register as shown in Tables 5-10 and 5-11.

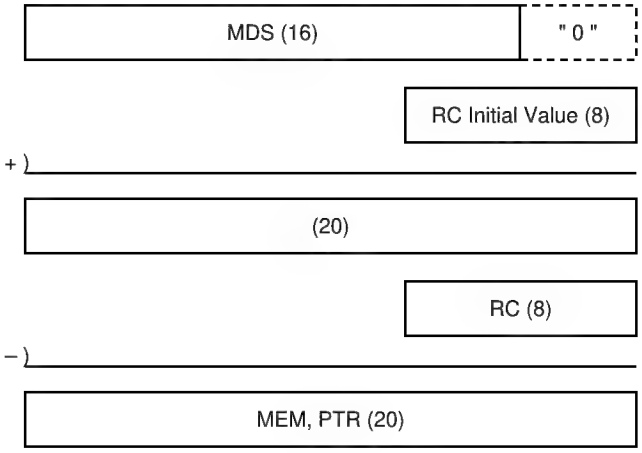
Table 5-10. Timer Compare Register Set Data Table Address Update Setting

U	Compare Register Set Data Reference Address
0	MEM.PTR (segment: MTS, offset: MTP) not updated
1	MEM.PTR – MSC × 2 + 2 (segment: MTS, offset: MTP) address updating

Table 5-11. Timer Compare Register Set Data Manipulation Specification

A	Compare Register Set Data
0	Set data in table transferred to compare register as it is
1	Set data in table + current compare register set data transferred

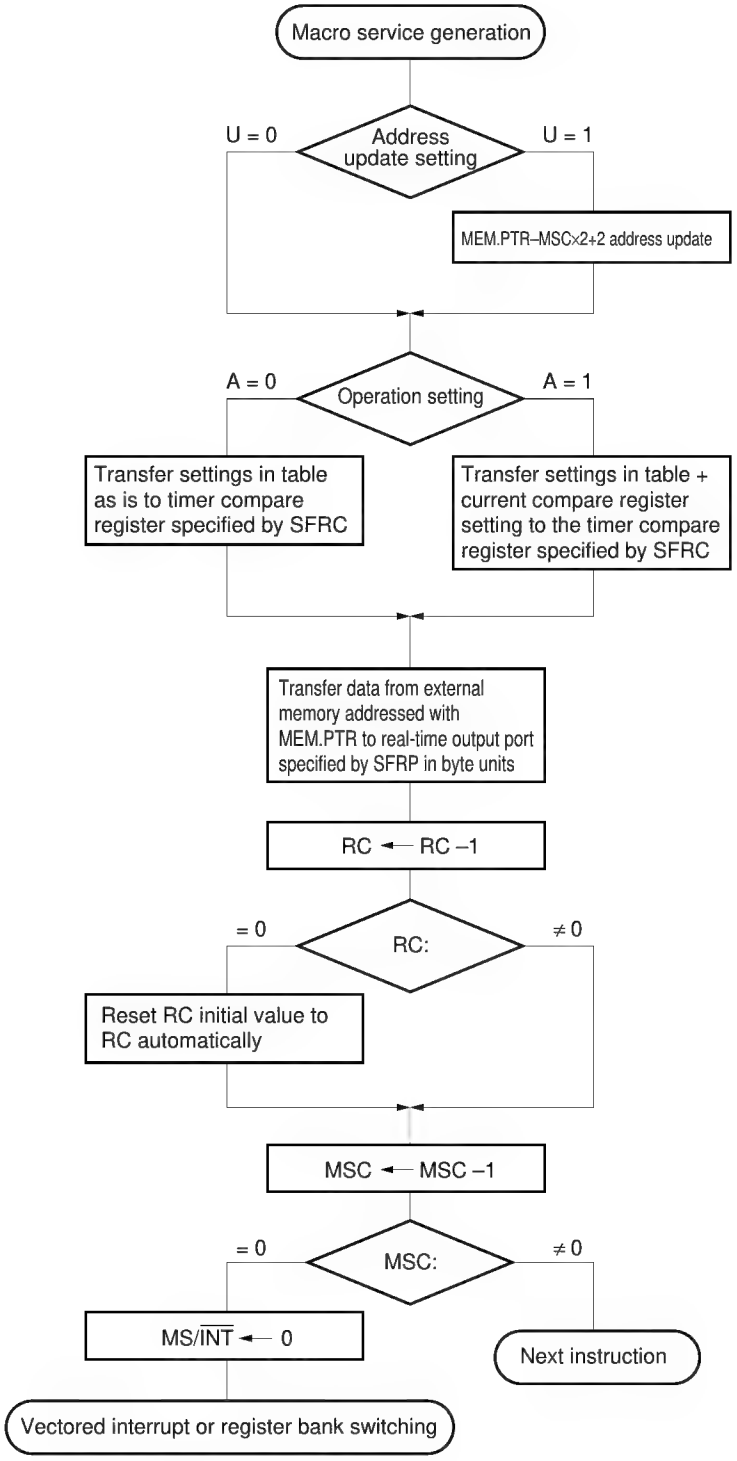
<2> The output data (byte data) is transferred to the real-time output port.
Data is transferred from the external memory addressed by MEM.PTR [memory segment (MDS) and memory pointer (ring counter initial value – ring counter)] to the real-time output port specified by SFRP in byte units, and the ring counter value is decremented (–1) (data can also be transferred to general-purpose port as well as a real-time output port).
Transfer object address (MEM, PTR) is obtained as follows. ( ) indicates the bit number.



<3> When the ring counter (RC) values reaches 0, the RC initial value is automatically set in the RC again.

<4> The macro service counter is decremented at each data transfer. When MSC = 0, the corresponding MS/$\overline{\text{INT}}$ of interrupt request control registers is set to 0 and a vectored interrupt or register bank switching is generated.

If 0 is set as the initial value of the MSC, transfer processing is performed 65536 times.
The low-order 8 bits of the respective SFR addresses are set to SFRC and SFRP.


Macro service generation
U = 0
Address update setting
U = 1
MEM.PTR–MSC×2+2 address update
A = 0
Operation setting
A = 1
Transfer settings in table as is to timer compare register specified by SFRC
Transfer settings in table + current compare register setting to the timer compare register specified by SFRC
Transfer data from external memory addressed with MEM.PTR to real-time output port specified by SFRP in byte units
RC ← RC –1
= 0
RC:
≠ 0
Reset RC initial value to RC automatically
MSC ← MSC –1
= 0
MSC:
≠ 0
MS/INT ← 0
Next instruction
Vectored interrupt or register bank switching

Cautions 1. The timer compare register and real-time output port involved in a macro service transfer are fixed as shown below for each interrupt request (INTCM00, INTCM01). This should be noted when setting the SFRC and SFRP.

Table 5-12. Interrupt Request Signals and SFRs

Interrupt Request Signal	Timer Compare Register	Port 7 Buffer
INTCM00	CM00	P7L
INTCM01	CM01	P7H

2. Writes to a real-time output port are as shown below according to the specification of the BYTE bit of the real-time output port control register (RTPC) (see CHAPTER 10 REAL-TIME OUTPUT PORT FUNCTION).

Table 5-13. Real-Time Output Port Write Specification

BYTE	Interrupt Request Signal	Write to Real-Time Output Port
0	INTCM00	Low-order 4 bits written to P7L
	INTCM01	High-order 4 bits written to P7H
1	INTCM01	Written to P7H and P7L in 8-bit units

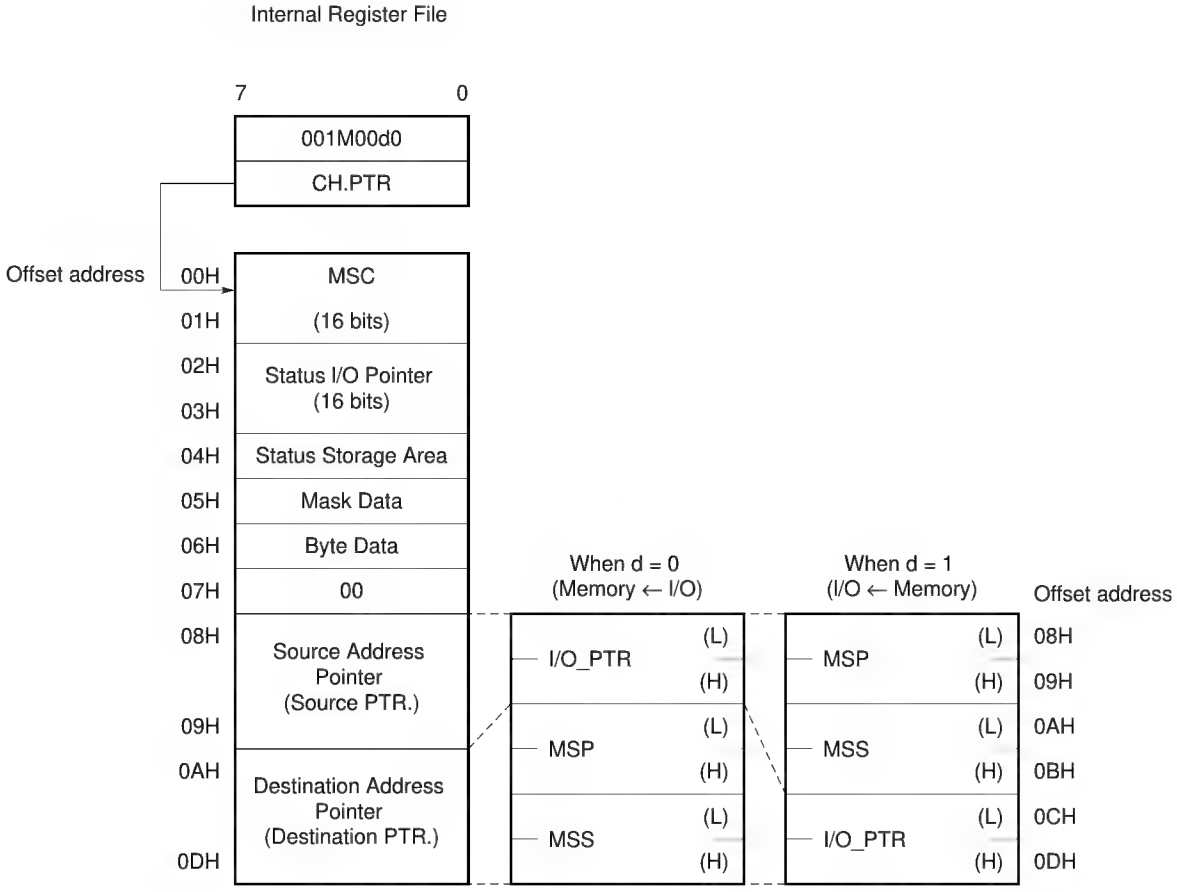
[Use]

Data transfer to the real-time output port 7 buffer and timer compare register (stepping motor control), etc. (See **Figure 10-4 Stepping Motor Control by Real-Time Output Port**)

(g) DTACMP-M (DATA COMPARE WITH CHARACTER MASK)

The logical product (AND) of the status data read from the external I/O and the preset mask data is obtained. The result is compared with the preset byte data. If they match, data is transferred between the external I/O and memory. If they do not match, or the preset number of data transfers has been performed, the macro service of the corresponding interrupt source is ended and a vectored interrupt or register bank switching is generated.

[Macro Service Control Word]



[Operation]

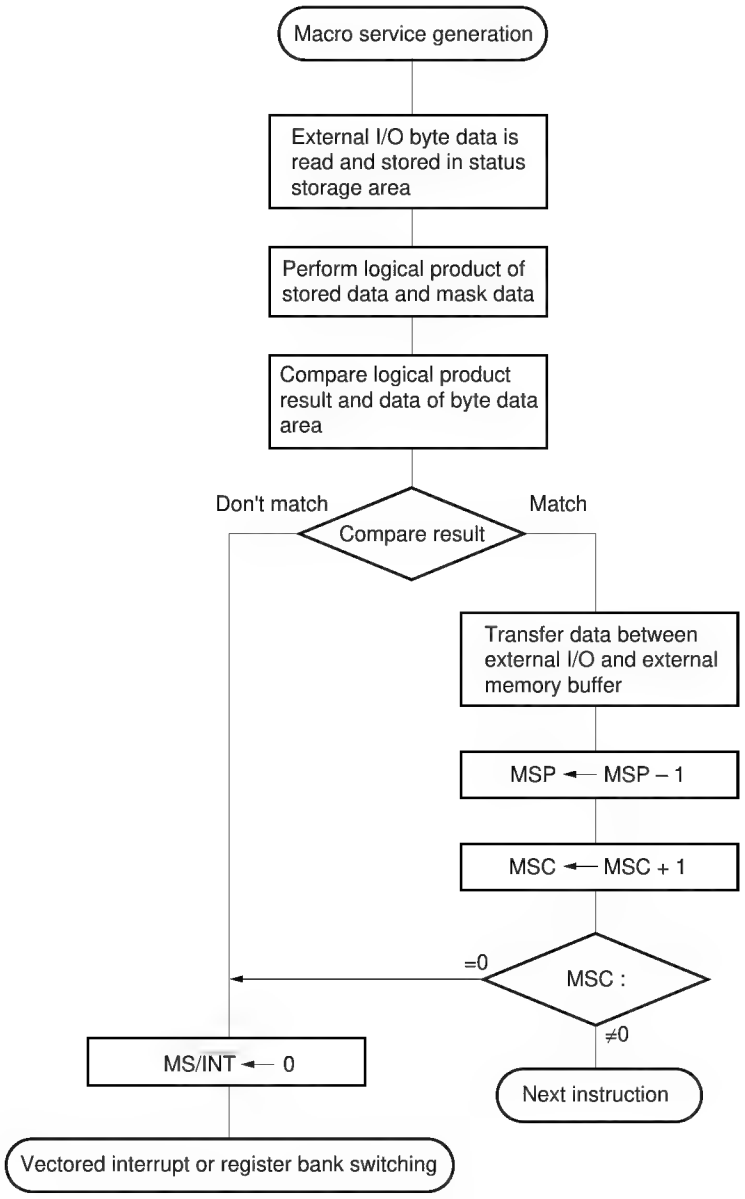
<1> The external I/O byte data of the address indicated by the status I/O pointer is read and stored in the status storage area.

<2> The logical product of the data stored in the status storage area and data stored in the mask data area is obtained. The result is compared with the data of the byte data area.

<3> If they match, data is transferred between the external I/O addressed by I/O_PTR and the external memory buffer addressed by MSP and MSS. If they do not match, the MS/$\overline{\text{INT}}$ bit of the corresponding interrupt request control register is cleared (0) and a vectored interrupt request or register bank switching request is generated.

<4> MSP increment and MSC decrement are performed for every data transfer. If MSC = 0 as a result of MSC decrement, the MS/$\overline{\text{INT}}$ bit of the corresponding interrupt request control register is cleared (0) and a vectored interrupt request or register bank switching request is generated.

Remark This macro service operation mode is for data transfer between the external I/O and memory. For details of transfer between SFR and memory, refer to **5.4.3 (6) (d) BLKTRS (BLOCK TRANSFER)**, **(e) BLKTRS-C (BLKTRS WITH CHARACTER SEARCH)**.



[Use]

Connected to modem, etc.

5.5 Control Registers

Interrupt control registers comprise the interrupt mode control register (IMC) which performs interrupt priority mode switching and vector table specification, interrupt mask flag registers (MK0, MK1) which control maskable interrupts, interrupt request control registers (INT××), and the in-service priority register (ISPR).

5.5.1 Interrupt mode control register (IMC)

This register performs priority control mode switching and vector table specification.

Data setting must be performed before using any interrupt function, including NMI.

The IMC register can be written to only once after a reset. If a subsequent rewrite is performed, the write data will be ignored.

The type of vector area used by the system (V55PI incorporated function) in the vector table area from 000000H and 0003FFH can be specified for the vector table by the V0 and V1 bits. Details of the vector table area are given in **3.5.4 Vector table area**.

See **5.3.3 Multiple interrupt control mode selection** for details of priority control modes set by the PRSL bit, and **5.12 Examples of Interrupt Request Service Operations** for concrete processing operations.

Caution Register setting should be performed even when a change from the initial state following a reset (value at time of reset = 80H) is not necessary. This serves to prevent an abnormal operation of the system due to subsequent miswrite operations.

	7	6	5	4	3	2	1	0
IMC	PRSL	0	V1	V0	0	0	0	0

Vector Table Specification

V1	V0	Vector Table Used by System	Address
0	0	8 to 47 (Initial value)	00020H-000BCH
0	1	72-111	00120H-001BCH
1	0	136-175	00220H:-02BCH
1	1	200-239	00320H-003BCH

Priority Control Mode Selection
(For level 3 maskable interrupts)

PRSL	Priority Control
0	Priority control not performed
1	Priority control performed

Caution To prevent misoperation in the interrupt sequence, a write to this register must only be performed in the interrupt disabled state. Therefore a DI instruction must always be executed before manipulating this register.

Remarks 1. The shaded area indicates status after a reset.

2. After a reset in the V55PI, vectors 8 to 47 function as an on-chip peripheral hardware system area, but the vector table area of vector 8 onward specification can easily be changed by means of the V0 and V1 bits of the IMC register. This allows easy migration of other V series application product programs.

Example V20/V30: Vectors 0 to 5 (CPU dedicated), vectors 6 to 31 (reserved area)
V25/V35: Vectors 0 to 7 (CPU dedicated), vectors 8 to 31 (reserved area)

5.5.2 Interrupt mask flag registers (MK0L, MK0H, MK1L, MK1H)

The interrupt mask flag registers specify service disabling/enabling for maskable interrupt requests corresponding to individual bits in the registers. The relationship between the individual mask flag numbers and interrupt requests is a correspondence to the default vector table numbers shown in Table 5-1. Individual flags can be manipulated bit-wise by bit manipulation instructions. After a reset the register value is FFH.

These flags are functionally and physically identical to the MK bits of the interrupt request control registers (IC××). That is, when the value of the interrupt request control register or the interrupt mask flag register is changed, the value of other register changes automatically (these registers can also be manipulated by means of a 16/ 8 or 1-bit unit memory access instruction).

	7	6	5	4	3	2	1	0
MK0L	1	MK14	MK13	MK12	MK11	MK10	MK09	1
MK0H	MK23	MK22	MK21	MK20	MK19	MK18	MK17	MK16
MK1L	MK31	MK30	MK29	MK28	MK27	MK26	MK25	MK24
MK1H	1	1	MK37	MK36	1	1	1	MK32

MK××	Interrupt Service Enable/Disable Specification
0	Enable interrupt service
1	Disable interrupt service

Remark The shaded area indicates status after a reset.

5.5.3 Interrupt request control registers (IC9 to IC14, IC16 to IC32, IC36 and IC37)

These registers set interrupt control for maskable interrupts on an individual source basis.

The value after a reset is 43H.

	7	6	5	4	3	2	1	0
ICxx	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0

Priority Specification

PR1	PR0	Priority to be Specified
0	0	Level 0
0	1	Level 1
1	0	Level 2
1	1	Level 3

Interrupt Service Mode Specification

MS/$\overline{\text{INT}}$	ENCS	Operation
0	0	Vectored interrupt servicing
0	1	Register bank switching processing
1	0	Macro service processing + post-termination vectored interrupt
1	1	Macro service processing + post-termination register bank switching

MS/$\overline{\text{INT}}$: Macro service processing flag
ENCS : Register bank switching processing flag

Interrupt Mask Specification

MK	Mask Specification
0	Not masked
1	Masked

Interrupt Response Indication

IF	Mask Specification
0	No interrupt request
1	Interrupt request present

Remark The shaded area indicates status after reset.

IF **Interrupt request flag**

The IF flag is set (1) when the corresponding interrupt request is generated, and cleared (0) automatically when the signal is acknowledged.

This flag is used to inititiate interrupt servicing. The IF flag can also be set/reset by software. Therefore, maskable interrupts can be generated by software. After a reset, it is initialized to "0".

MK **Interrupt mask flag**

This is functionally identical to the flag in the interrupt mask flag registers (MK0, MK1) (when either is manipulated, the other corresponding flag is also rewritten with the same contents).

After a reset, it is initialized to "1", and interrupt disabled state is set.

MS/$\overline{\text{INT}}$ **Macro service processing bit**

This bit specifies that interrupt requests are to be serviced by macro service.

After a reset, it is initialized to "0" (vectored interrupt service specification).

ENCS **Register bank switching processing bit**

This bit specifies that interrupt requests are to be serviced by register bank switching.

After a reset, it is iniatialized to "0" (vectored interrupt service specification).

PR1 **PR0** **Priority specification bits**

These bits are used in multiple interrupt servicing: Maskable interrupt priorities can be set in 4 stages from level 0 to level 3 (with level 0 as the highest interrupt priority level).

After a reset, these bits are initialized to "11", specifying the lowest priority level.

Remark Only vectored interrupts and maskable interrupts serviced by register bank switching are affected by the contents of PR1 and PR0. Macro service is not affected by the contents of these bits. Processing is always performed as a highest-level interrupt (see **5.12 Examples of Interrupt Request Service Operations**).

The correspondence between the interrupt request control registers (IC××) and interrupt source is shown below.

IC09	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP0)
IC10	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP1)
IC11	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP2)
IC12	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP3)
IC13	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP4)
IC14	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTP5)
IC16	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM00)
IC17	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM01)
IC18	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM10)
IC19	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM11)
IC20	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM21)
IC21	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTCM31)
IC22	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTD1 DMA#0_MAIN)
IC23	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTD1S DMA#0_SUB)
IC24	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTD1 DMA#1_MAIN)
IC25	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTD1S DMA#1_SUB)
IC26 Note	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTSER0)
IC27 Note	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTSER1)
IC28	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTSR0/INTCSI0)
IC29	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTSR1/INTCSI1)
IC30	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTST0)
IC31	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTST1)
IC32 Note	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTSIT)
IC36	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTPAI)
IC37	IF	MK	MS/$\overline{\text{INT}}$	ENCS	0	0	PR1	PR0	(INTAD)

Note Ensure that 0 is set in MS/$\overline{\text{INT}}$.

5.5.4 In-service priority register (ISPR)

This register records and holds the priority level of the maskable interrupt service currently being serviced. It is used when the interrupt controller performs priority control.

When an interrupt request is acknowledged, the bit corresponding to the priority level is set in order. When multiple interrupts are subsequently acknowledged, the corresponding bit is set (1) each time.

Bit resetting is performed by the FINT instruction. The bit in the ISPR register corresponding to the highest priority is cleared automatically.

	7	6	5	4	3	2	1	0
ISPR	0	0	0	0	Level 3	Level 2	Level 1	Level 0

Priority Interrupt Service Status

0	Interrupt request not being acknowledged
1	Interrupt request being acknowledged

Cautions 1. When an RETI or RETRBI instruction is used to return from a maskable interrupt service routine, an FINT instruction must always be executed immediately before the return instruction.

2. An FINT instruction must not be executed when returning from a non-maskable interrupt or sofware interrupt service routine.

Remark The shaded area indicates status after reset.

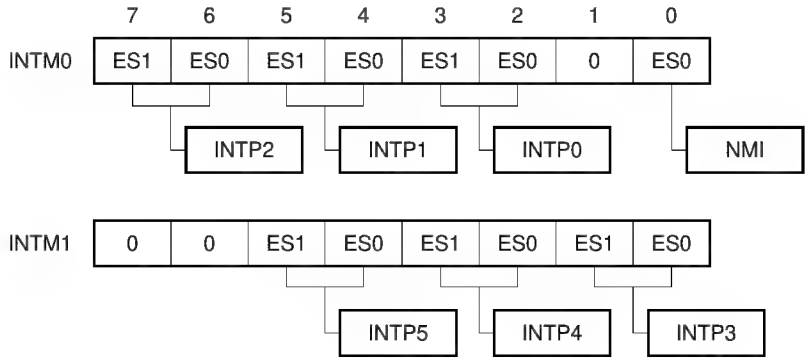
5.6 External Interrupts

There are 7 external interrupt requests: NMI and INTP0 to INTP5.

5.6.1 External interrupt mode registers (INTM0, INTM1)

The valid edge can be selected for external interrupt acknowledged. This can be specified by the external interrupt mode registers (INTM0, INTM1) (special function registers).

After a reset, the value of the INTM0 and INTM1 registers is 00H.



Valid edge specification

ES1	ES0	Valid Edge
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Rising and falling edges

Caution "Rising and falling edges" cannot be specified for NMI.

Remark The shaded area indicates status after a reset.

The external interrupt pins (INTP0 to INTP5) are also used for trigger signal generation for other on-chip peripheral hardware units (timer unit, A/D converter, parallel interface controller).

If not used as external interrupt pins, these pins must be masked by the interrupt mask flag register. In particular, alternate trigger function disabling/enabling control is required for the INTP3, INTP4, and INTP5 pins.

5.7 Software Interrupts

The V55PI has a total of 11 interrupts generated by software (see **Table 5-14**).

These are compatible with V25 and V35 software-generated interrupts.

Software interrupts are not subject to control by the on-chip interrupt controller, and their interrupt vectors are predetermined.

Except for the interrupt generated by the BRK flag (single-step interrupt), these interrupts are always acknowledged when the condition for their generation is met (they have priority over hardware interrupts). Also in the case of contention between an NMI and a software interrupt, the software interrupt has priority.

However, an interrupt due to the BRK flag is generated when BRK = 1 without distinction between hardware and software, but when the interrupt is acknowledged, the BRK flag is reset automatically, and thus its priority becomes lower than other interrupts (both hardware and software), and an interrupt due to the BRK flag is not generated during servicing of that interrupt.

Table 5-14. Software Interrupts

Interrupt Source	Vector	Priority
DIVU divide error	0	1
DIV divide error		
BRK flag (single-step)	1	2
BRK3	3	1
BRKV	4	
CHKIND boundary exceeded	5	
Input/output instruction ($\overline{\text{IBRK}}$ flag)	6	
BRK imm8	**Note**	
BRKCS instruction	–	
FPO instruction	7	
Exception trap		

Note Any value between 0 and 255 (0 and FFH) (specified by imm8)

5.7.1 General software interrupts

The acknowledgement sequence for software-generated interrupts other than those generated by input/output instructions and those generated by an FPO instruction is the same as for vectored interrupts. That is, address information for the next instruction and the PSW are saved to the stack, IE and BRK are set to 0, and the vector contents are loaded into PS and PC.

Software interrupts are described below.

(1) DIVU divide error, DIV divide error

Always generated when the quotient resulting from execution of a division instruction overflows.

(2) CHKIND boundary exceeded

Generated when the index value is determined to exceed the boundary as the result of executing an instruction (CHKIND) which checks whether or not the index value exceeds the predefined array boundary.

(3) BRKV

Generated if V (the overflow flag) is set (1) when the BRKV instruction is executed.

(4) BRK 3

Generated by execution of the BRK 3 instruction.

(5) BRK imm8

Generated by execution of the BRK imm8 instruction.

(6) BRK flag (single step)

Generated each time an instruction is executed when BRK is set (1).
If a repeat prefix is added, this interrupt is not generated until the loop is exited.

5.7.2 Input/output instruction interrupts

An interrupt is generated if an attempt is made to execute an input/output instruction when $\overline{IBRK} = 0$. **The address information saved to the stack when this interrupt is acknowledged differs from that for a general software interrupt (see 5.7.1 General software interrupts), consisting of the address at which that input/output instruction is located.** If a prefix is added to the input/output instruction, the address is that at which the prefix is located. Otherwise, the operation is the same as for a general software interrupt. When returning from an input/output instruction interrupt, the value of the PC saved on the stack must be adjusted to effect a normal return.

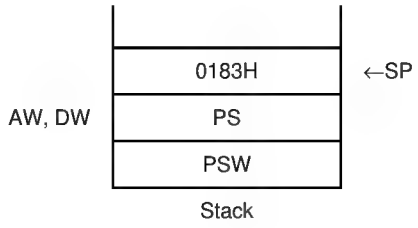
Using the start address of the instruction as the address information saved to the stack makes it possible to know by attempted execution of which instruction by software the interrupt was generated. This function facilitates migration of programs which use an external I/O used on the previous V20 and V30.

The contents of the PSW directly before the interrupt is generated are saved to the stack, after which IE and BRK are set to 0 and $\overline{IBRK}$ to 1 automatically. As a result of setting $\overline{IBRK}$ to 1, the instruction for which interrupt service is in progress is executed as an input/output instruction and the original state ($\overline{IBRK} = 0$) is restored automatically on returning from the interrupt.

Example 1. Without prefix

Address instruction

0183H

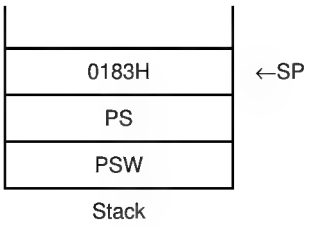


Example 2. With prefix

Address instruction

0183H REP

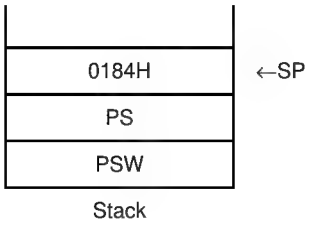
0814H INM



Reference General software interrupt

Address instruction

0183H BRK 3



5.7.3 FPO instruction interrupt

Because the external bus configuration of the V55PI is different from that of the V20 and V30, a V20/V30 floating-point coprocessor cannot be connected to it. Therefore, if an attempt is made to execute a coprocessor FPO instruction, an interrupt is generated to emulate the operation of that instruction. **With this interrupt, as with an input/output instruction interrupt, the PC value saved to the stack is the start address of that instruction (or the prefix start address if a prefix is added) (See 5.7.2 Input/output instruction interrupts).** It is therefore possible to decode the FPO instruction and perform emulation by software. As with an input/output instruction interrupt, the value of the PC saved on the stack must be adjusted when returning from an FPO instruction interrupt in order to effect a normal return.

5.8 Exception Trap

An exception trap is generated by an illegal write to the watch timer mode register (WDM).

This interrupt is not provided in the V25 or V35: It is a new function accompanying the provision of a watchdog timer in the V55PI.

The exception trap interrupt is serviced using vector 7, and shares the same vector number as the FPO interrupt.

For information on write instructions to the WDM register, see **12.2 Configuration and Operation**.

5.9 Times at which Hardware Interrupt Requests are not Acknowledged

Harware interrupt requests are not acknowledged between the instructions shown below and the following instruction.

(1) Segment register sreg and extended segement register xsreg manipulation instructions
MOV sreg, reg16 ; MOV sreg, mem16 ; POP sreg
MOV xsreg, reg16; MOV xsreg, mem16; POP xsreg

(2) Program status word (PSW) manipulation instruction
POP PSW

(3) Prefix instructions
PS:, SS:, DS0:, DS1:, DS2:, DS3:, IRAM:,
REPC, REPNC, REP, REPE, REPZ, REPNE, REPNZ,
BUSLOCK

(4) EI, DI, RETI, RETRBI

(5) FINT

(6) BRKCS, BRK3, BRKimm8,
BRKV (when V = 1)
CHKIND (when mem32 > reg16 or (mem32 + 2) < reg16)
FPO1 fp-op, FPO1 fp-op, mem
FPO2 fp-op, FPO2 fp-op, mem,
TSKSW, MOVSPB, MOVSPA
DIV, DIVU (when divide error)

(7) RSTWDT

A hardware interrupt request generated when it cannot be acknowledged is acknowleged after execution of the next instruction if interrupts can be acknowledged at that time.

When the hardware interrupt request is acknowleged, other hardware interrupt requests are acknowledged after the EI instruction is executed during the interrupt service routine.

5.10 Interrupt Servicing during Block Processing Instruction Execution

Interrupt requests are acknowledged during execution of a block processing instruction (transfer, comparison, search, input/output).

An interrupt request is acknowledged when one data transfer has been completed. At this time, the address saved in the stack or register bank PC save area automatically becomes the start of instructions including a prefix. When returning from the interrupt, the block processing which was being executed can be resumed by re-executing from the prefix.

Up to 4 kinds of prefix including a repeat prefix can theoretically be added to block processing instructions. In the VP55PI, it is possible to determine what prefix is attached to the block processing instruction being executed when an interrupt is acknowledged, and the PC value is decremented accordingly before being saved to the stack.

Example REP
MOVBK SS:SRC_BLK, DES_BLOCK

5.11 Interrupt Servicing during Codec Instruction Execution

An interrupt request is acknowledged even in execution of an instruction for the following 6 of the 9 codec instructions.

Instructions	Operands	Instruction Processing Unit
Change point table creation instruction	COLTRP	One-word pixel data processing of change point table
MH encoding instruction	MHENC	One-word encoding processing of change point table
MR encoding instruction	MRENC	MR encoding processing in 1-mode units (one of path mode, vertical mode, and horizontal mode)
MH decoding change point table creation instruction	MHDEC	One-word decoding processing of change point table
MR decoding change point table creation instruction	MRDEC	MR decoding processing in 1-mode units (one of path mode, vertical mode, and horizontal mode)
Pixel data creation instruction	CNVTRP	One-word pixel data conversion processing of change point table

An interrupt request is sampled each time the one-time instruction processing is completed. If an interrupt request is ready to be acknowledged with mask specification and priority control, execution of a codec instruction is aborted and the system branches to interrupt servicing. Then, the start address of the codec instruction being executed (including added prefixes) is saved to the stack or register bank PC save area.

The parameters being used by the aborted codec instruction (on the register or register bank) are consecutively used at the time of restoration from interrupt servicing. However, the same status must be kept as the original at the time of restoration from interrupt servicing. Therefore, do not manipulate data during interrupt servicing.

The following is the non-sampling time (maximum value) of each instruction interrupt request in execution of A4 size (1728 bits/line) processing with external data bus of 16 bits and with no wait.

(Unit: μs)

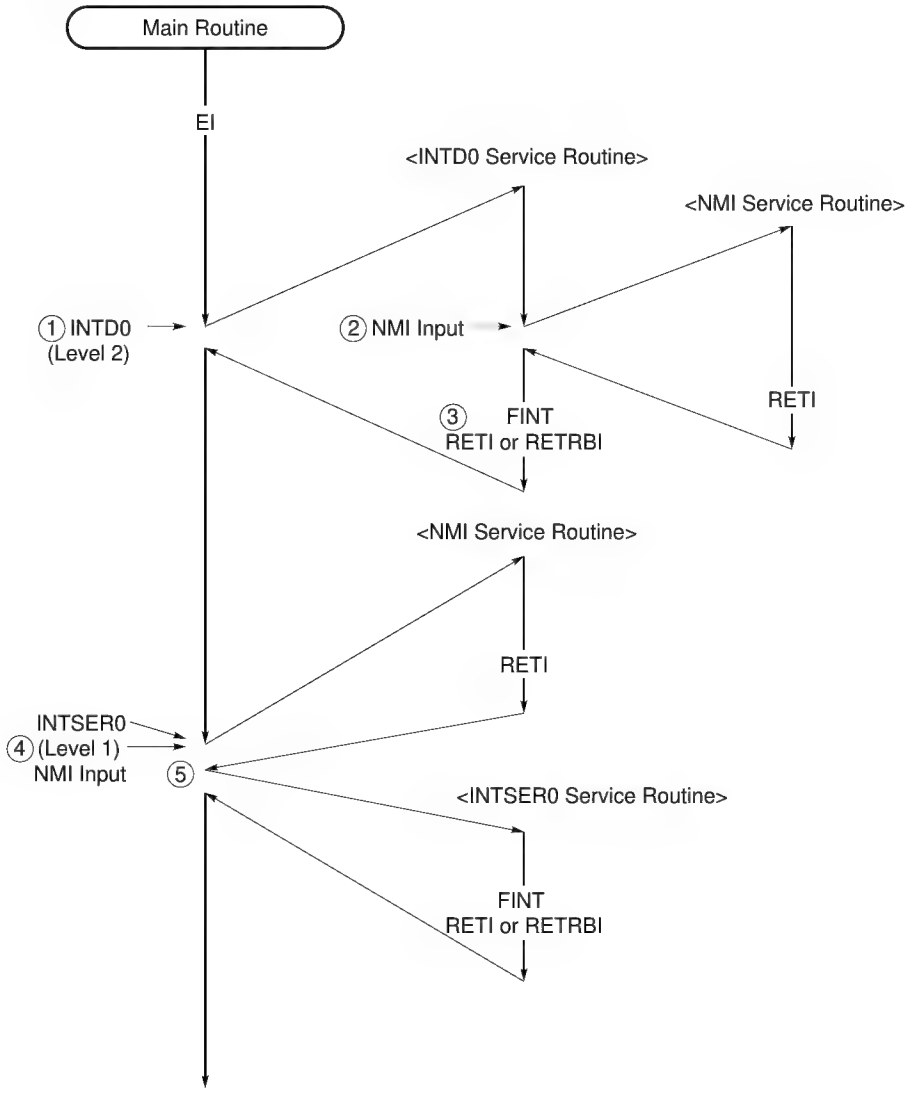
Processing	Method	COLTRP	MHENC	MRENC	MHDEC	MRDEC	CNVTRP
Encoding	MH	4.6	19	–	–	–	–
	MR	4.6	19	24	–	–	–
Decoding	MH	–	–	–	23.52	–	58.16
	MR	–	–	–	23.52	33	58.16

When interrupt requests are generated while instructions (ALBIT, SCHEOL, GETBIT) which do not perform interrupt sampling are executed, one request is reserved for every interrupt source at the most. The reserved interrupt is acknowledged if the state is interrupt enabled (EI) after the codec instruction ends.

5.12 Examples of Interrupt Request Service Operations

(1) NMI

- Not masked by software
- Has a higher priority for acknowledgement than any other interrupt.



① An INTD0 interrupt set as priority level 2 is generated in the interrupt enabled state (EI), and INTD0 servicing is started.

② The interrupt disabled state (DI) is set during INTD0 servicing, but as a result of NMI generation INTD0 servicing is suspended and NMI servicing is started.

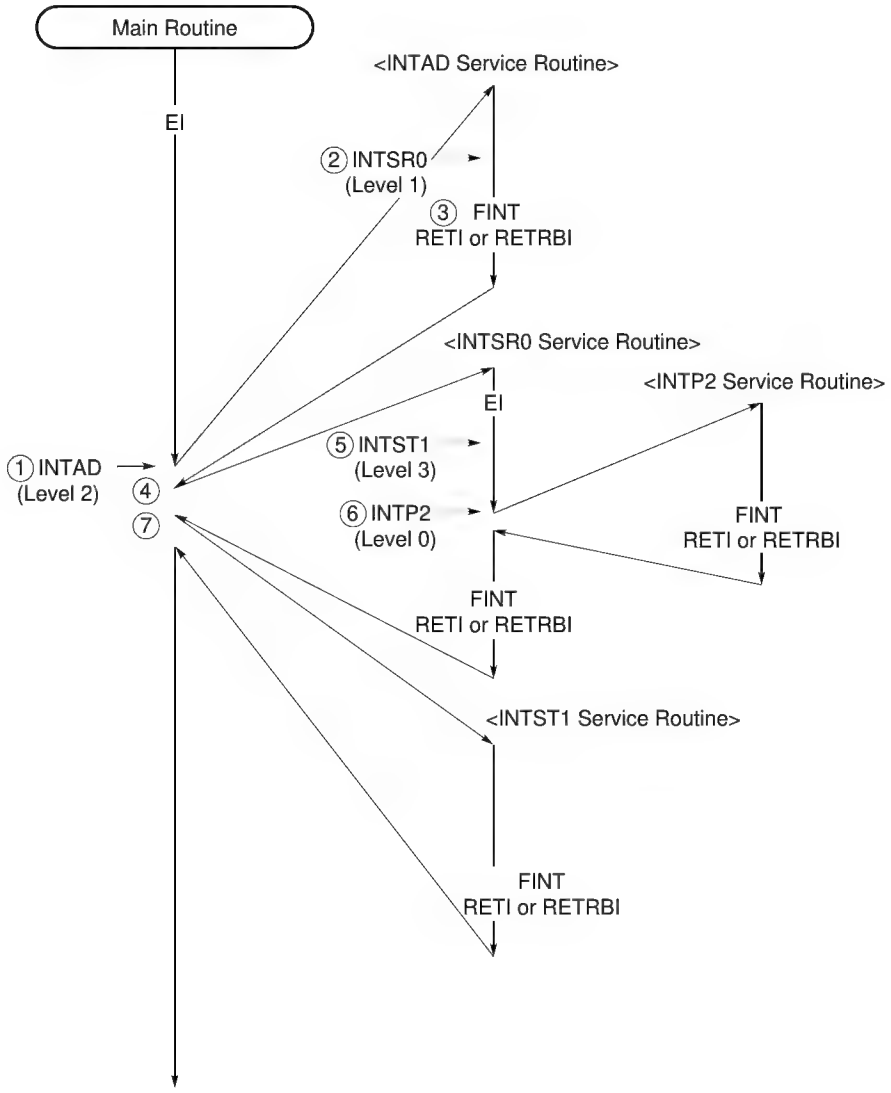
③ When NMI servicing ends, the suspended INTD0 interrupt is serviced and a return is effected by executing a return instruction.

④ When an INTSER0 interrupt set as priority level 1 is generated at the same time as an NMI, the NMI is acknowledged (as it has priority over all other interrupts) and NMI servicing is started.

⑤ When NMI servicing ends, the pending INTSER0 is serviced.

(2) Interrupts subject to multiple interrupt servicing control ..... A

- Multiple servicing is performed according to priorities.



① An INTAD interrupt set as priority level 2 is generated in the interrupt enabled state (EI), and INTAD servicing is started.

② Because the interrupt disabled state (DI) is set when INTAD is acknowledged, although an INTSR0 interrupt set as priority level 1 (higher than INTAD) is generated, it is not acknowledged.

③ For interrupts subjected to priority control, an FINT instruction must be executed when a return is effected.

④ When INTAD servicing ends and the interrupt enabled state is set, servicing of the pending INTSR0 is started.

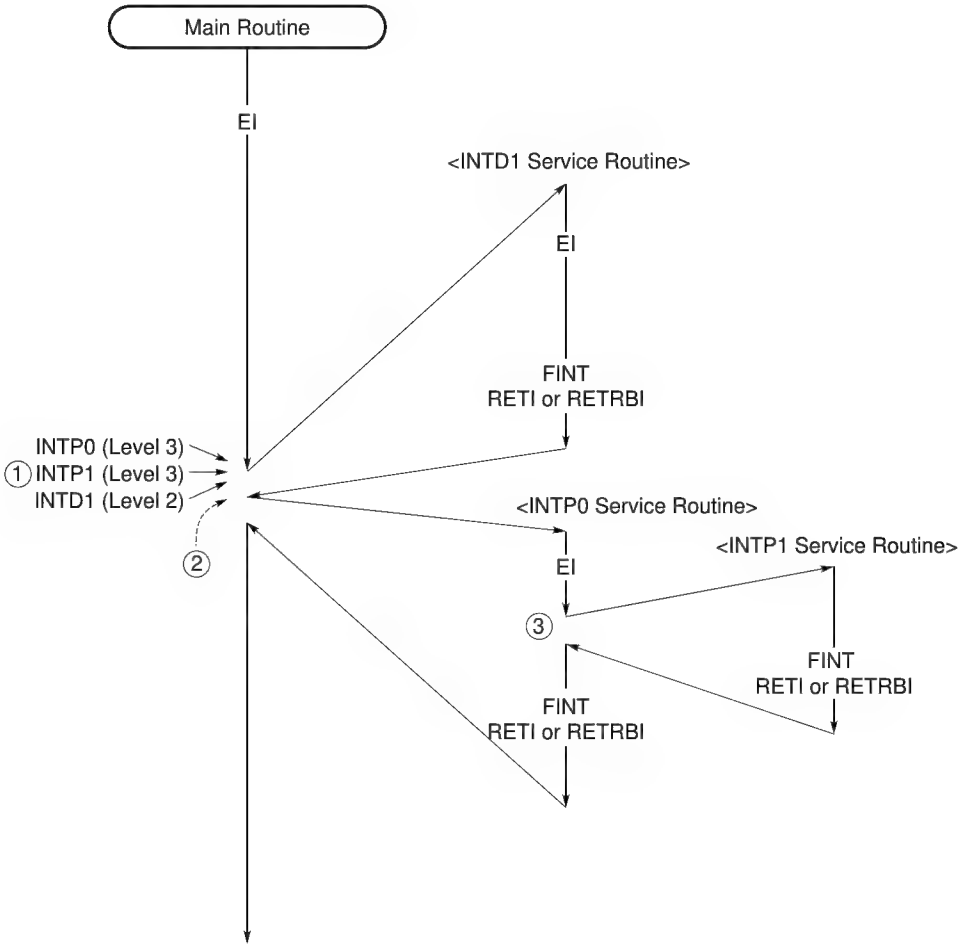
⑤ An INTST1 interrupt set as priority level 3 is generated in the interrupt enabled state (EI), but INTSR0 servicing is in progress and its priority is higher than that of the INTST1 interrupt, INTST1 is not acknowledged.

⑥ The INTP2 interrupt set as priority level 0 generated next is acknowledged because it has higher priority than INTSR0, and INTP2 servicing is started.

⑦ When INTP2 servicing ends, the suspended INTSR0 interrupt service is ended and a return is effected, then INTST1 is acknowledged.

(3) Interrupt subject to multiple interrupt servicing control ..... B (PRSL = 0)

- When the PRSL bit is set to 0, priority 3 maskable interrupts are not subject to priority control and same-level nesting is possible.

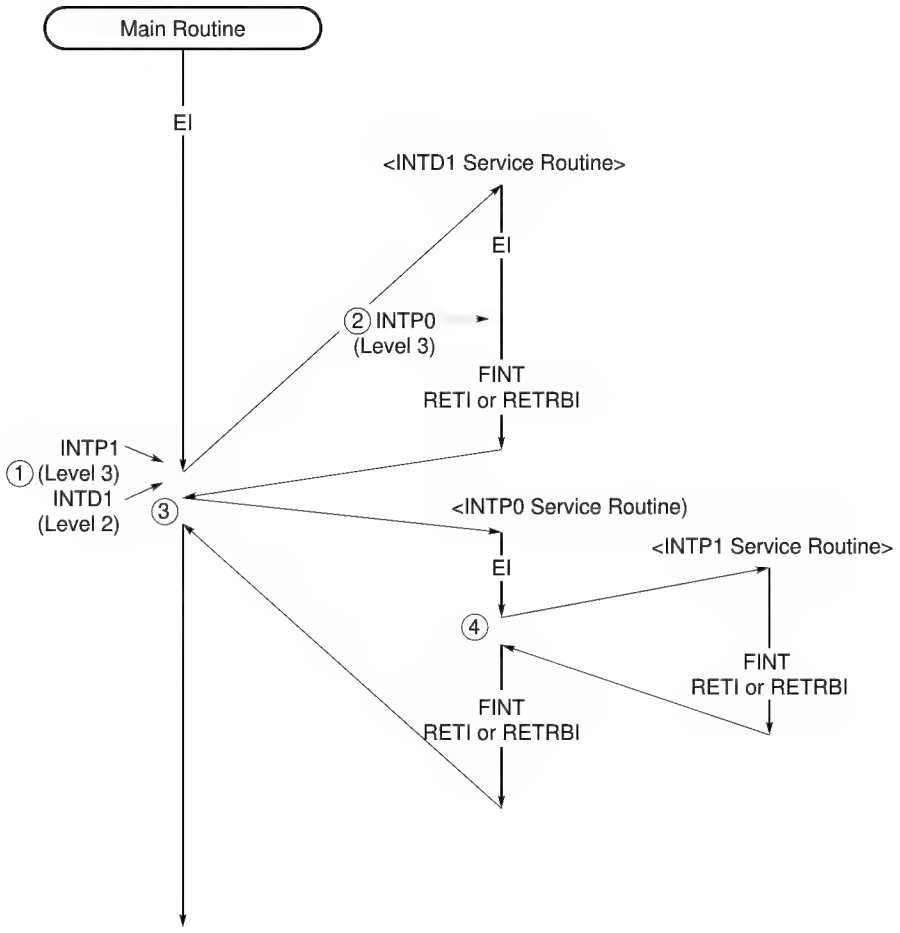


① When INTP0 and INTP1 set as priority level 3 and INTD1 set as priority level 2 are generated simultaneously in the interrupt enabled state (EI), INTD1 with a higher priority set by the program is acknowledged and INTD1 servicing is started.

② When INTD1 servicing ends, since the pending INTP0 and INTP1 interrupts have the same priority level set, INTP0, which has the higher default priority, is acknowledged.

③ When the interrupt enabled state (EI) is set, servicing of the pending INTP1 interrupt is started.

(4) Interrupts subject to multiple interrupt servicing control ..... C (PRSL = 0)



① When INTP1 set as priority level 3 and INTD1 set as priority level 2 are generated simultaneously in the interrupt enabled state (EI), INTD1 with a higher priority set by the program is acknowledged and INTD1 servicing is started.

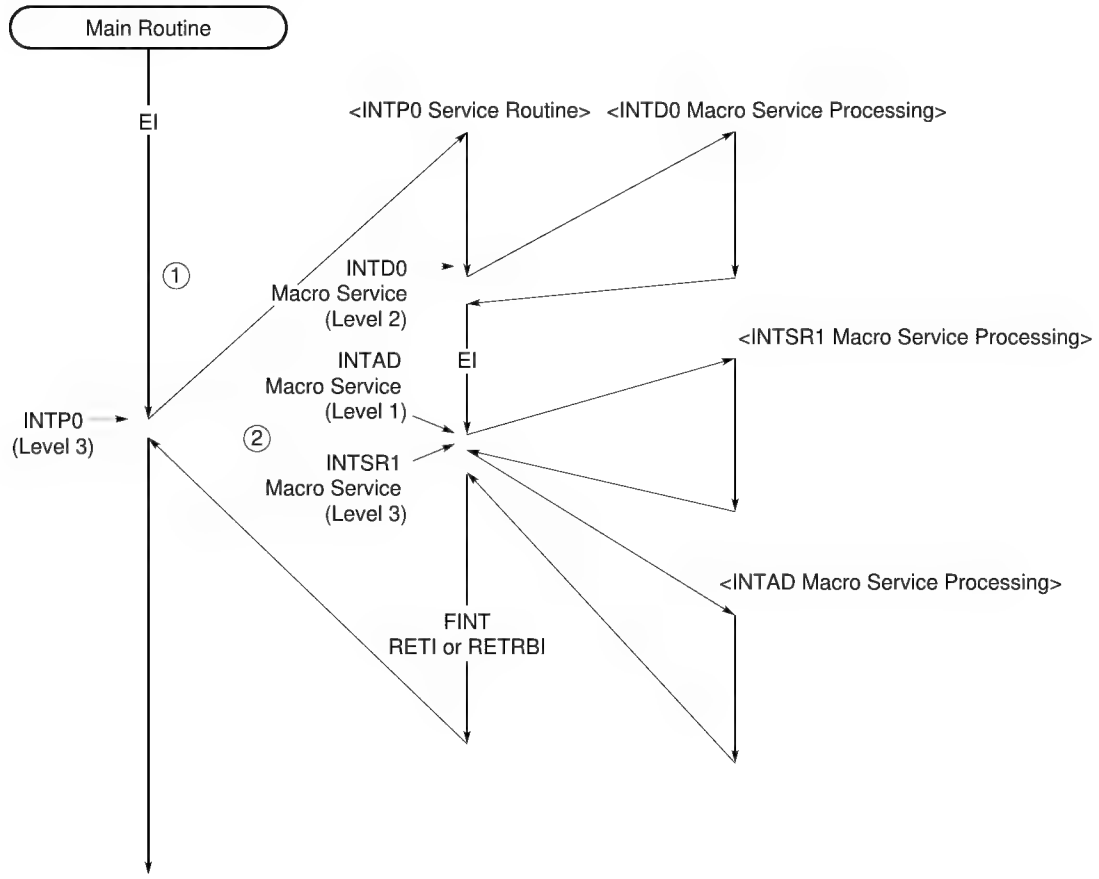
② An INTP0 interrupt set as priority level 3 is generated in the interrupt enabled state (EI), but INTD1 servicing is in progress and INTD1 priority is higher than that of the INTP0 interrupt, INTP0 is not acknowledged.

③ When INTD1 servicing ends, because the pending INTP0 and INTP1 interrupts have the same priority level set, INTP0, which has the higher default priority, is acknowledged.

④ When the interrupt enabled state (EI) is set, servicing of the pending INTP1 interrupt is started.

(5) Macro service interrupts ..... A

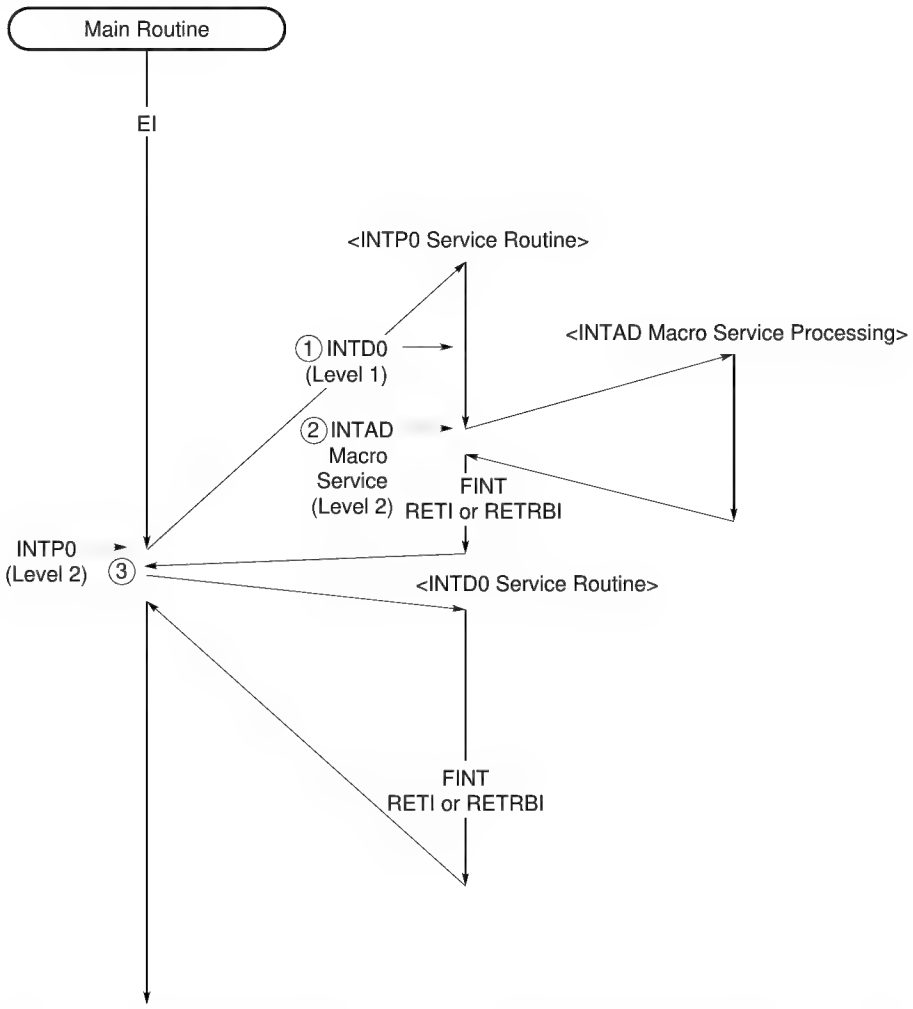
- Acknowledged irrespective of whether the interrupt enabled (EI) or interrupt disabled (DI) state is in effect.
- Not subject to PR flag priority specification control.



① When an INTD0 macro service interrupt set as priority level 2 is generated during servicing of an INTP0 interrupt set as priority level 3, macro service processing is performed even through the interrupt disabled state (DI) is in effect.

② Next, when INTAD and INTSR1 macro service interrupts set as priority level 1 and 3 respectively are generated simultaneously during servicing of the INTP0 interrupt and in the interrupt enabled state (EI), they are acknowledged in order starting with INTSR1 macro service interrupt which has the higher default priority.

(6) Macro service interrupts ..... B



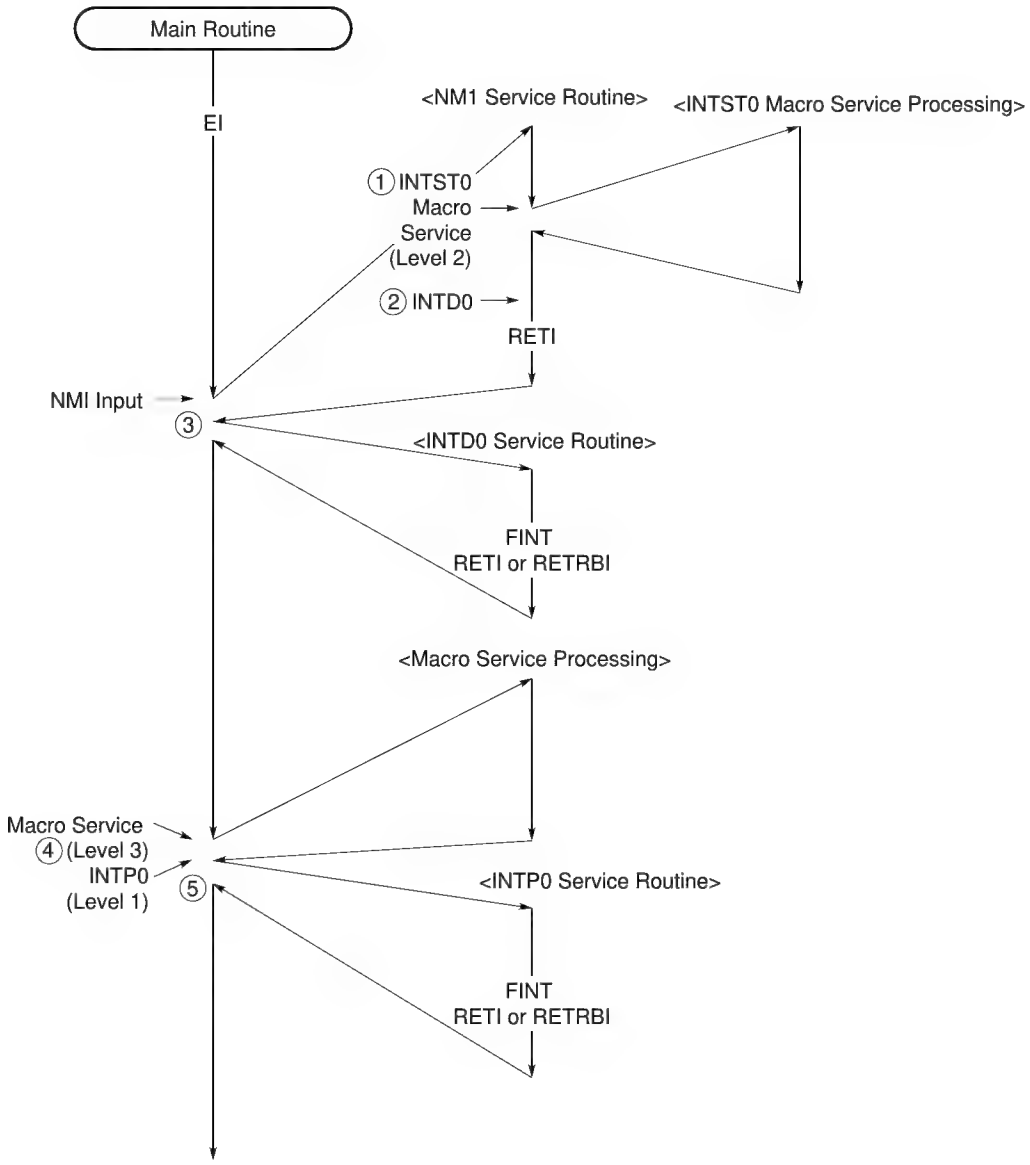
① An INTD0 interrupt request set as priority level 1 is generated during servicing of an INTP0 interrupt set as priority level 2, but because the interrupt disabled state (DI) is in effect, it is held pending.

② When an INTAD macro service interrupt request set at priority level 2 is generated, macro service processing is performed irrespective of the priority of the interrupt being serviced and the pending interrupt, or the interrupt disabled state (DI).

③ When INTP0 servicing ends, the pending INTD0 interrupt is acknowleged and INTD0 servicing is started.

(7) Priority of macro service interrupts with respect to other interrupts

- Macro service interrupts are acknowledged even during NMI servicing



① When a macro service interrupt is generated, macro service processing is peformed even though NMI servicing is in progress and the interrupt disabled state (DI) is in effect.

② When an interrupt (INTD0) subject to multiple servicing control is generated while NMI servicing is in progress and the interrupt disabled state (DI) is in effect, it is not acknowleged.

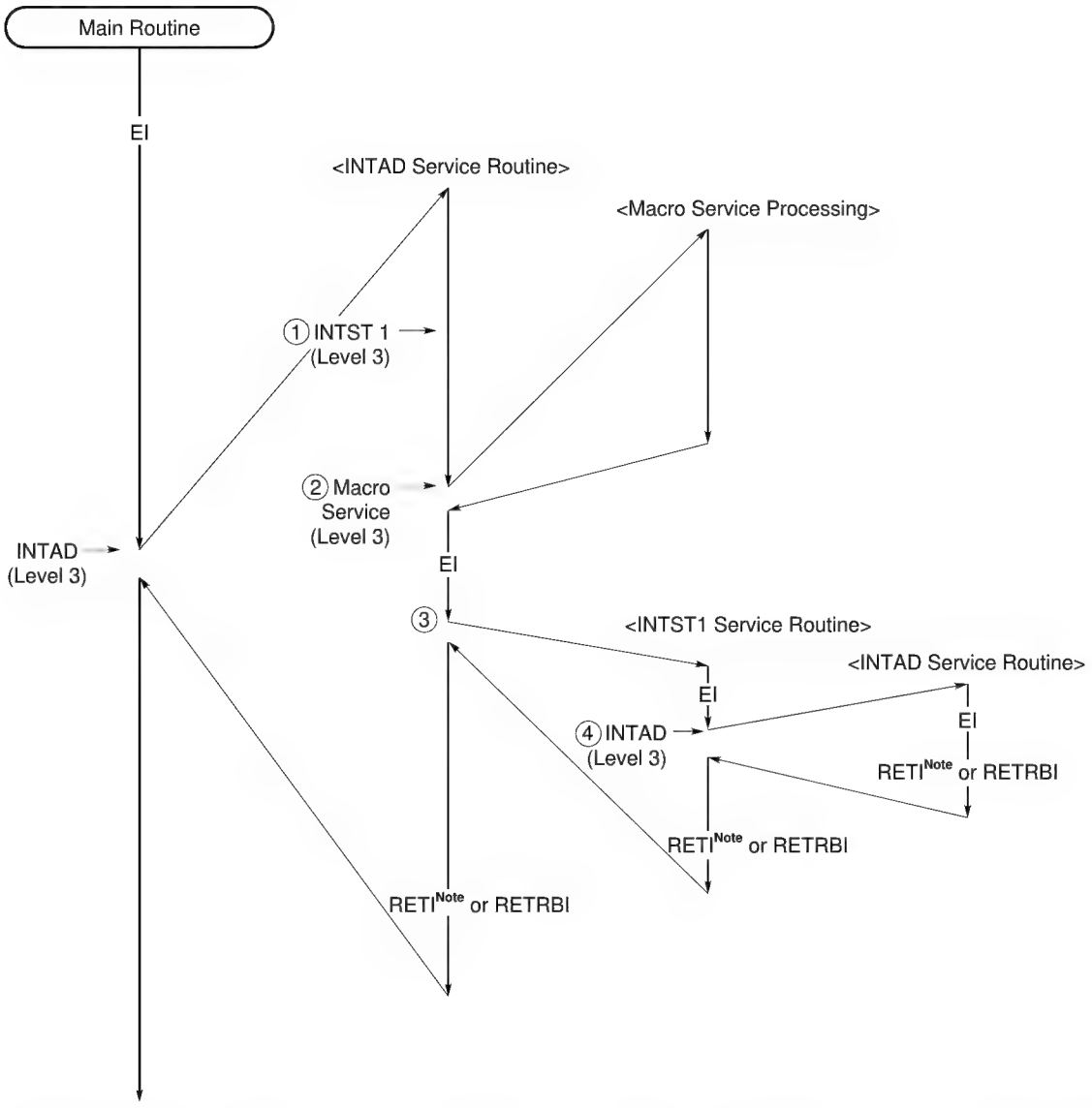
③ When NMI servicing ends, the pending INTD0 interrupt is acknowledged.

④ When a macro service interrupt and an interrupt (INTP0) subject to multiple servicing control are generated simultaneously in the interrupt enabled state (EI), the macro service interrupt is acknowleged first, and INTP0 is held pending.

⑤ When macro service processing ends, the pending INTP0 interrupt is acknowledged.

(8) Multiple interrupt servicing (PRSL = 0)

- Interrupts subject to multiple servicing control, macro service interrupts



① Because an INTAD interrupt set as priority level 3 is acknowledged and the interrupt disabled state (DI) is set, when an INTST1 interrupt set as priority level 3 is generated, it is not acknowledged.

② When a macro service interrupt set as priority level 3 is generated, macro service processing is performed.

③ When the interrupt enabled state (EI) is set, servicing of the pending INTST1 interrupt is started.

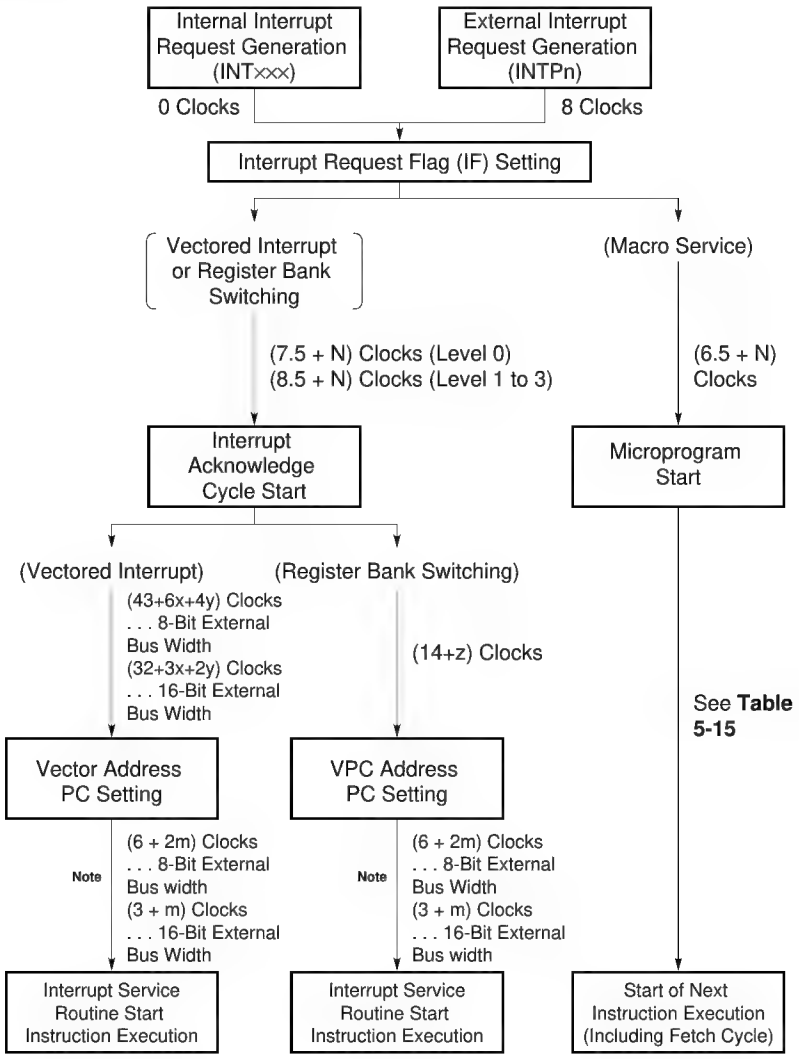
④ If the interrupt enabled state (EI) is in effect during INTS1 servicing, when an INTAD interrupt is generated, INTAD servicing is started. If it is not wished to perform multiple servicing of identical interrupts, the mask flag should be set (1) in the interrupt service routine before setting the interrupt enabled state.

Note When PRSL = 0, because priority 3 interrupt generation is not recorded in the interrupt controller's ISPR register, it is not necessary to execute an FINT instruction (also, issuing an FINT instruction will not cause any particular problems in terms of operation).

5.13 Interrupt Response Time

The following is the interrupt response time when such as hold request, DMA transfer request, and refresh request are not considered.

(1) Maskable interrupt



N : Number of remaining execution clock cycles of an instruction being executed by CPU at this time
x : Number of waits for memory when PC, PS, and PSW are saved to stack
y : Number of waits for memory when vector PC and vector PS are read
z : Number of waits for memory when register bank number is read from vector table
m : Number of waits for memory when start instruction of interrupt service routine is fetched (execution of instruction is started when 2 bytes of instruction are fetched)

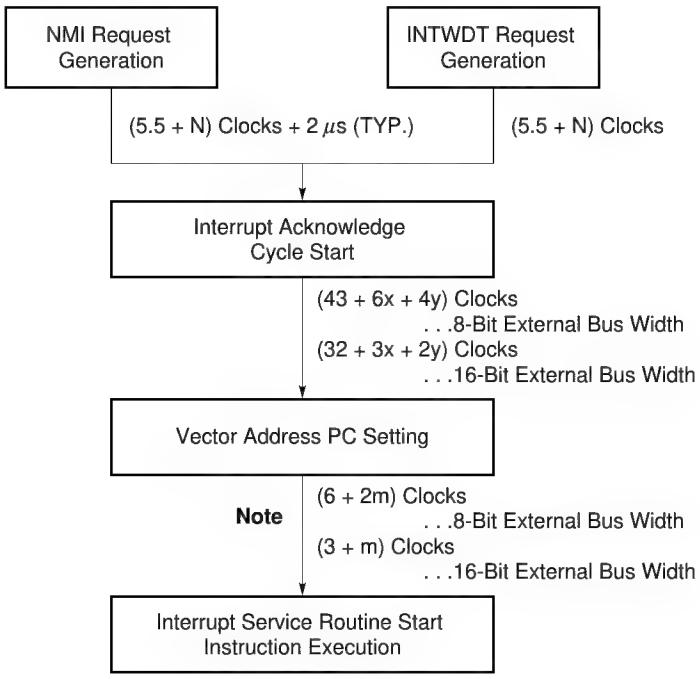
Note This number of clocks applies when start instruction of interrupt service routine is fetched.

Table 5-15. Macro Service Processing Time (Unit: Clock)

Macro Service Function			8 Bits Bus Width	16 Bits Bus Width
EVTCNT	INC mode		15	15
	DEC mode		15	15
DTACMP			22	22
DTADIF	Byte processing		29 + t	29 + t
	Word processing		33 + t	30 + t
BLKTRS	Memory ← SFR	Byte processing	29 + 2t	29 + 2t
		Word processing	35 + 4t	31 + 2t
	SFR ← Memory	Byte processing	30 + 2t	30 + 2t
		Word processing	36 + 4t	31 + 2t
BLKTRS-C (MSC ≠ 0)	Memory ← SFR	Byte processing	40 + t	40 + t
		Word processing	42 + t	40 + t
	SFR ← Memory	Byte processing	41 + t	41 + t
		Word processing	49 + t	46 + t
BLKTRS-C (MSC = 0)	Memory ← SFR	Byte processing	30 + t	30 + t
		Word processing	33 + t	30 + t
	SFR ← Memory	Byte processing	33 + t	33 + t
		Word processing	36 + t	34 + t
RTOPTRN	A = 0		75 + 4t	72 + 2t
	A = 1		79 + 4t	76 + 2t
DTACMP-M	When aborted by status determination		27 + m	27 + m
	When performing external I/O → memory transfer		51 + m + n	51 + m + n
	When performing memory → external I/O transfer		49 + m + p	49 + m + p

t : Number of waits for transfer memory
m : Number of read cycle waits for status data (I/O)
n : Number of read cycle waits for transfer data (I/O)
p : Number of read cycle waits for transfer data (memory)

(2) Non-maskable interrupt



N : Number of remaining execution clock cycles of an instruction being executed by CPU at this time
x : Number of waits for memory when PC, PS, and PSW are saved to stack
y : Number of waits for memory when vector PC and vector PS are read
m : Number of waits for memory when start instruction of interrupt service routine is fetched (execution of instruction is started when 2 bytes of instruction are fetched)

Note This number of clock cycles applies when start instruction of interrupt service routine is fetched.

5.14 Caution

5.14.1 When performing SFR access by macro service

Set the low-order 16 bits of the SFR address to the SFR address specification pointer of the macro service channel. Set 1 to all the high-order 7 bits.

5.14.2 Interrupt mode control register manipulation

Always set the interrupt mode control register once even if the initial state after reset need not be changed, to prevent abnormal system operation caused by incorrect writing operations in future.

Always perform this register manipulation in the interrupt disabled state (DI state).

5.14.3 Interrupt request generation after macro service ends

When the same interrupt request is generated during the vectored interrupt servicing (or register bank switching) which generates when macro service ends, the IF flag of the interrupt request control register becomes 1. When the MS/$\overline{\text{INT}}$ bit of the interrupt request control register is changed from 0 to 1, macro service is executed immediately. Interrupt service is executed in succession when it ends.

5.14.4 If an interrupt is not generated

If no interrupt is generated when it should have been, first check the following points.

(1) Is the status EI?

Check whether the EI instruction has been executed. If the IE flag of the PSW register has been set (to "1"), the V55PI is currently in the EI status.

(2) Is the interrupt masked?

In this case, reset (to "0") bit 6 of the interrupt request control register (IC××) to cancel the mask.

(3) Has an interrupt request been generated?

If bit 7 (IF) of the interrupt request control register (IC××) is set (to "1"), an interrupt request is generated. When this bit is "0", no interrupt request is generated. Check whether the interrupt request has been cleared by software.

(4) Is a high-priority interrupt being processed?

A bit corresponding to a high interrupt priority may be set to the in-service priority register (ISPR). In this case, the V55PI checks whether an interrupt of the current priority level of the set bit is currently being processed, and will not accept an interrupt with a lower priority level.

The bits of the ISPR register can be set only with the FINT instruction. Therefore, check whether the FINT instruction is being executed during interrupt processing of the set priority level.

For details on the ISPR register, see **In-service priority register (ISPR)**.

5.14.5 Cautions regarding FINT instruction

The FINT instruction must be executed immediately before the last RETI instruction during interrupt processing.

The FINT instruction clears the bit corresponding to the highest priority level of the bit set in the ISPR register. If the FINT instruction is not executed during the interrupt service routine, interrupts with a priority level lower than that interrupt are not acknowledged.

Also, if the FINT instruction is executed while an interrupt is being processed, the V55PI becomes unable to control priorities, and starts accepting also interrupts with a priority level lower than that of the interrupt currently being processed. Therefore, if the FINT instruction is executed while an interrupt is being processed, control priority levels from the application.

For details on the FINT instruction, see the manual **V55PI UM Instructions**.

CHAPTER 6 DMA FUNCTION (DMA CONTROLLER)

The V55PI incorporates a 2-channel DMA controller which controls execution of DMA transfers on the basis of memory-to-I/O or memory-memory DMA (Direct Memory Access) requests generated by an on-chip peripheral hardware (serial interface, parallel interface, or timer), the external DMARQ pin, or a software trigger.

Each channel of the DMA controller further comprises a main channel and a sub-channel: the operating mode determines whether the main channel and sub-channel are used as a single channel or as separate channels. Using the channels separately constitutes a DMA function equivalent to that of a maximum of 4 channels.

6.1 Features

- Two independent DMA channels (max. 4-channel configuration possible)
- Four transfer modes
  - Single transfer mode ... One DMA transfer cycle is executed for one DMA request.
  - Demand release mode ... DMA transfer cycle is executed consecutively while DMA request is active.
  - Single step mode ... After generation of DMA request, DMA transfer cycle and CPU bus cycle are executed alternately.
  - Burst mode ... For one DMA request, DMA transfer cycle is executed consecutively for the specified number of times.
- 5 kinds of operating mode
  - Intelligent DMA mode-1 (ring buffer method) ... Control of DMA transfer for ring buffer
  - Intelligent DMA mode-2 (counter control method) ... Consecutive transfer of transfer data divided into arbitrary byte lengths
  - Next address specification mode ... Consecutive transfer between different transfer buffers is possible.
  - 2-channel operating mode ... Main channel and subchannel are used independently.
  - Memory to memory transfer mode ... 2 bus cycles are started for one-DMA transfer cycle and memory-to-memory transfer is executed.
- 3 clocks/1 bus cycle (with no wait)
- Transfer target
  - Between external I/O and memory ... 1 DMA transfer cycle/1 bus cycle
  - Between SFR (internal I/O) and memory ... 1 DMA transfer cycle/1 bus cycle
  - Memory to memory (memory includes SFR) ... 1 DMA transfer cycle/2 bus cycles
- Byte transfer/word transfer selectable
- Transfer address increment/decrement/non-update selectable
- 24-bit DMA memory address registers (MAR0, MAR1)
- 21-bit terminal counters (TC0, TC1)
- DMA controller control pin (See **2.2 Description of Pin Functions**)
  - External DMA request signal input pins (DMARQ0, DMARQ1: alternate function as port P80 and P81 pins)
  - External DMA acknowledge signal output pins ($\overline{\text{DMAAK0}}$, $\overline{\text{DMAAK1}}$)
  - DMA transfer end signal output pins ($\overline{\text{TCE0}}$, $\overline{\text{TCE1}}$)

6.2 Basic Operation

By using the DMA function, data transfer between memories, and between I/O (including special function register (SFR)) and memory can be realized without the CPU. The DMA function has the following advantages.

- Data transfer rate is significantly improved by hardware control.
- Response to external events is better than that of the software processing by interrupt.
- The overhead in a program processing is basically only the bus operation, so reduction of the processing capability of the CPU caused by data transfer is minimized.
- As a simple data transfer processing does not require user programs, programs are efficiently coded.

Two types of transfer operations are available — I/O (partially includes SFR)-to-memory, memory (partially includes SFR)-to-memory. Their transfer methods differ as follows.

Table 6-1. DMA Transfer Target and Transfer Method

Transfer Operation	Transfer Target	Transfer Method	Transfer Source Selection	Transfer Target Selection
I/O-to-memory transfer	External I/O → External memory	A write operation to a transfer source and a read operation to a transfer target are executed in one bus cycle.	$\overline{\text{DMAAK}}$ output	Memory address
	External I/O ← External memory		Memory address	$\overline{\text{DMAAK}}$ output
	SFR[Note 1] → External memory		Internal dedicated signal	Memory address
	SFR[Note 1] ← External memory		Memory address	Internal dedicated signal
Memory-to-memory transfer[Note 2]	External memory → External memory	A read operation to a transfer source is executed in the first bus cycle and a write operation to a transfer target is executed in the second bus cycle.	Memory address	Memory address
	SFR → External memory		SFR address	Memory address
	SFR ← External memory		Memory address	SFR address

Notes 1. Only PAB, RxB0, RxB1, TxB0, and TxB1 can be specified.

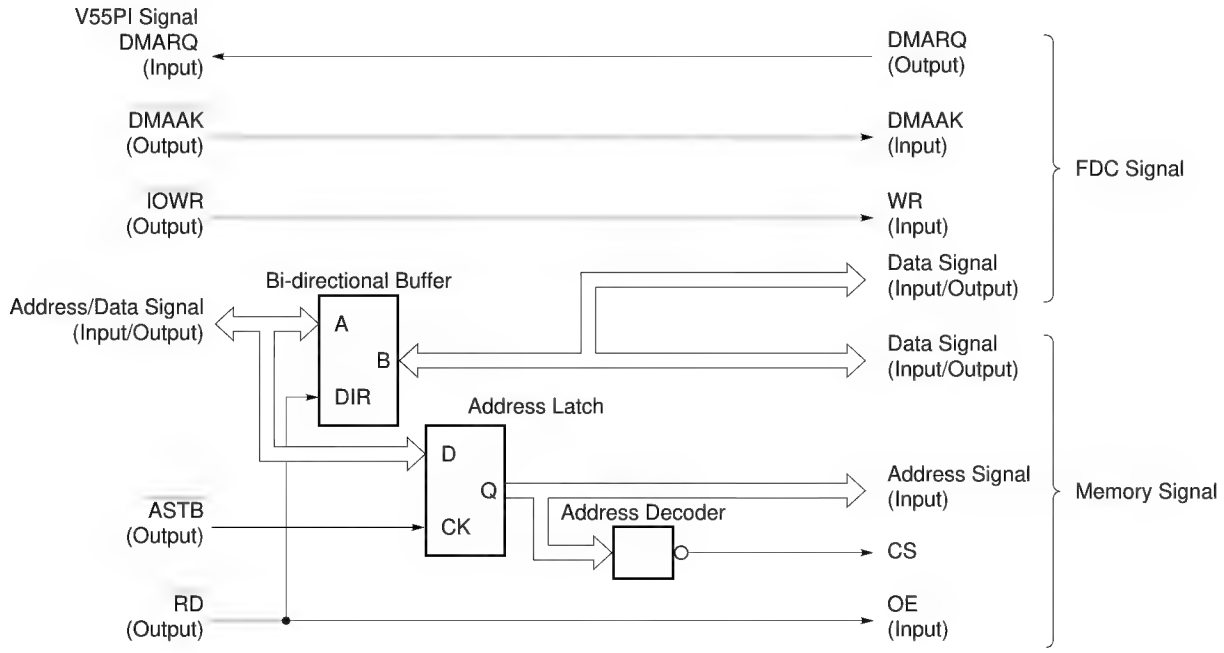
2. When the memory-to-memory transfer which manages the SFR is executed, the demand release mode cannot be used.
This is because a bus cycle is not generated during SFR accessing and the number of transfer cannot be controlled.

When DMA transfer between the 8-bit SFR and memory is executed when the external data bus width is 16 bits, the valid data bus is as follows:

- When external memory address is even : lower 8 bits (D0 to D7)
- When external memory address is odd : higher 8 bits (D8 to D15)

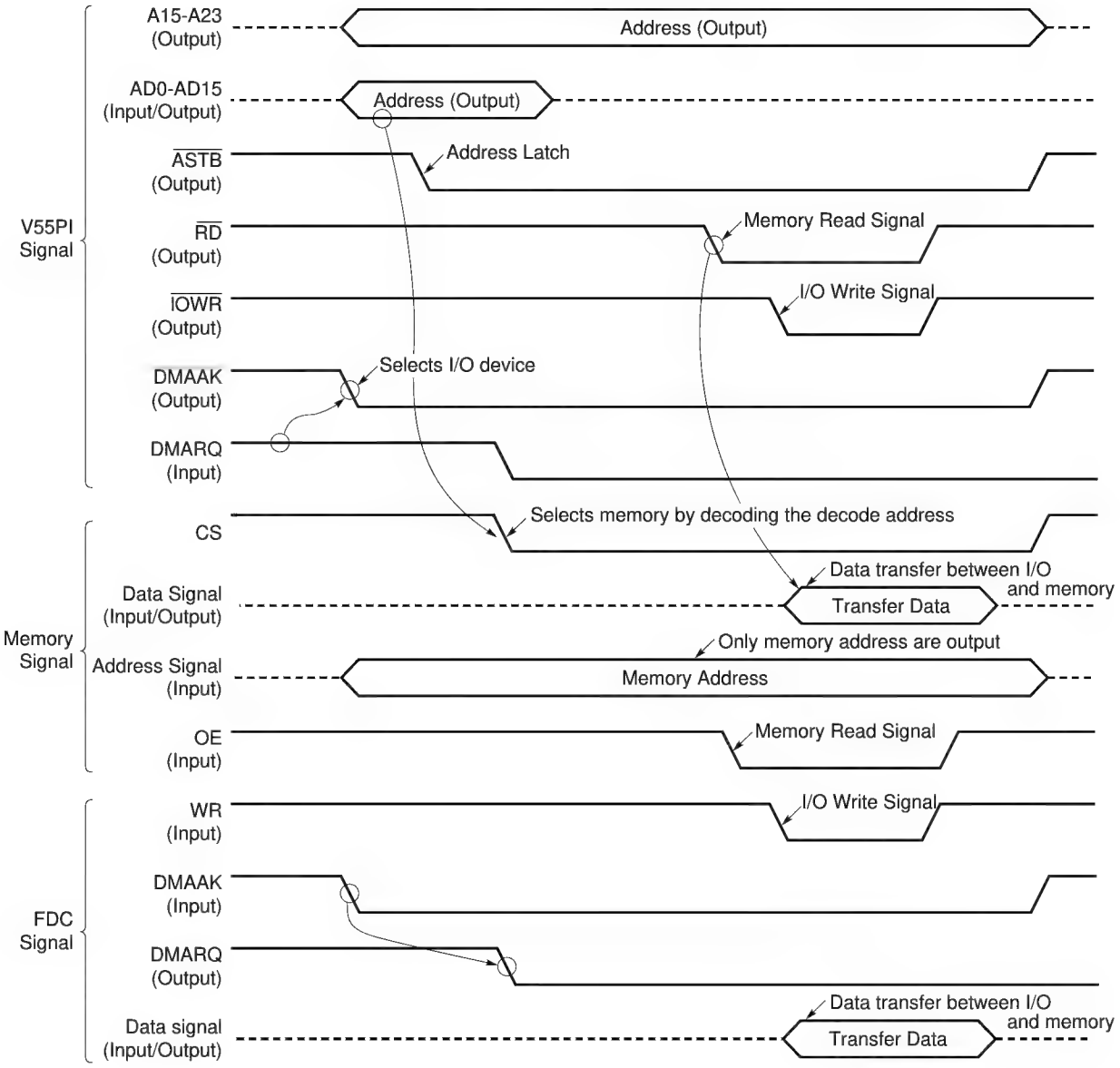
Figure 6-1 shows a connection example of $\overline{\text{DMAAK}}$ output when an FDC (Floppy Disk Controller) of the μPD72070 equivalent is connected, and Figure 6-2 shows its timing example. They are examples of an external memory to external I/O transfer in the single transfer mode.

Figure 6-1. $\overline{\text{DMAAK}}$ Output Example (Single Transfer Mode)
(Memory → I/O Transfer)



Remark This circuit example shows signals used only for DMA transfer cycle.
Timing should be considered when designing the actual circuit.

Figure 6-2. DMA Transfer Timing Example (Single Transfer Mode)
(Memory → I/O Transfer)



Remark The broken line indicates high-impedance.

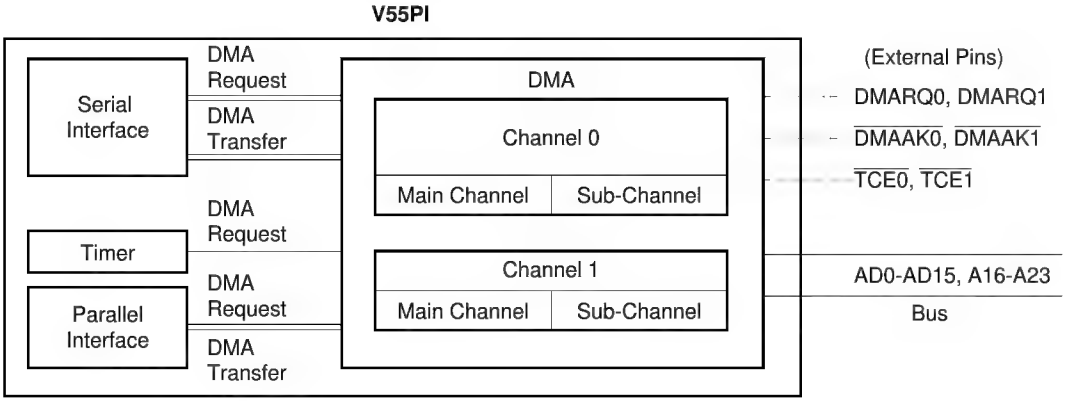
6.3 Configuration

The DMA controller of the V55PI has a pair of independent DMA channels. In addition to external memory-to-external memory/external I/O-to-external memory DMA transfer that are possible with conventional DMA controllers, this controller can manage SFR of on-chip peripheral hardware-to-external memory DMA transfer. Additionally, this controller has an intelligent DMA mode that can automatically perform, besides data transfer, pointer control of transfer target memory buffers.

6.3.1 Channel configuration

The channel configuration of the DMA controller is shown in Figure 6-3. The pair of DMA channels consists of a main channel and sub-channel respectively. When the 2-channel operating mode is specified, a maximum of 4 DMA channels can be used because channels can operate independently. However, the combination of transfer target and start source is limited (See **Table 6-2 Possible Combination of Transfer Target and Start Source**).

Figure 6-3. Channel Configuration



6.3.2 Transfer target

The following transfer targets (transfer source and transfer target) can be selected for the DMA data transfer.

Using the MD bit and TS bit of the DMA mode register (DMAM0 and DMAM1), the combination of the operating mode and start source is specified. Transfer direction is specified with the DIR bit of the DMAC register.

(1) External memory (Consecutive 2 Mbytes (max.) in a 16-Mbyte space can be specified.)
(2) External I/O
(3) SFR
- Parallel interface buffer (PAB)
- UART transmit buffer (TxB0, TxB1)
- Receive buffer (RxB0, RxB1)
- Port (P0 to P8), Port 7 buffer (P7L, P7H)
- Real time output port (RTP)
- Timer compare register (CM00, CM01, CM10, CM11, CM20, CM21, CM22, CM23, CM30, CM31)
- PWM register (PWM)
- A/D conversion result register (ADCR0 to ADCR3)

Caution When the DMA function and macro service are used together, do not specify the same memory block or SFR as the transfer target for both processes.

6.3.3 Start source

The following start sources can be selected for the DMA transfer cycle. Responding to transfer requests from these start sources, the DMA transfer is executed. Using the TS bit of the DMA mode register (DMAM0 and DMAM1) and the transfer target in combination specifies start source.

(1) External pin input (DMARQ0 and DMARQ1)
(2) Internal interrupt request
    - Parallel interface interrupt (INTPA1)
    - UART transmit interrupt (INTST0 and INTST1)
    - UART receive interrupt (INTSR0 and INTSR1)
(3) Match signal of timer/counter compare register (CM23)

By manipulating the TDMA bit of the DMAM register, the DMA transfer cycle can be activated using software.

Table 6-2. Possible Combination of Transfer Target and Start Source (1/2)

(Channel 0)

Corresponding Transfer Mode (Transfer Object)	TS1	TS0	Main Channel		Sub-Channel[Note 1]	
			Start Source	Transfer Target I/O	Start Source	Transfer Target I/O
I/O-to-memory transfer	0	0	Parallel interface INTPAI	Data input/output buffer (PAB)	Serial transmission completion INTST0	Transmit buffer (TxB0)
			I/O ↔ Memory		Memory → I/O	
			DIR = 0 or 1[Note 2]		Reverse direction to main channel	
	0	1	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)
			I/O ↔ Memory		I/O ↔ Memory	
			DIR = 0 or 1		Same direction as main channel	
	1	0	On-chip timer CM23 match signal	External I/O ($\overline{\text{DMAAK0}}$)	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)
			I/O ↔ Memory		I/O ↔ Memory	
			DIR = 0 or 1		Reverse direction as main channel	
	1	1	Serial transmission completion INTST0	Transmit buffer (TxB0)	Serial transmission completion INTST1	Transmit buffer (TxB1)
			Memory → I/O		Memory → I/O	
			DIR = 1 (fixed)		Same direction as main channel	
Memory-to-memory transfer	0	0	Setting prohibited			
	0	1	DMARQ0	–		
	1	0	On-chip timer CM23 match signal	–		
	1	1	Setting prohibited			

Notes 1. The sub-channel is only valid in the 2-channel operating mode.

2. If the value set in DIR is 1 (memory → I/O), and the parallel interface is selected as the start source, the sub-channel side cannot be used.

Table 6-2. Possible Combination of Transfer Target and Start Source (2/2)

(Channel 1)

Corresponding Transfer Mode (Transfer Object)	TS1	TS0	Main Channel		Sub-Channel[Note 1]	
			Start Source	Transfer Target I/O	Start Source	Transfer Target I/O
I/O-to-memory transfer	0	0	Serial reception completion INTSR1	Receive buffer (RxB1)	Serial transmission completion INTST0	Transmit buffer (TxB0)
			I/O → Memory		Memory → I/O	
			DIR = 0		Reverse direction to main channel	
	0	1	On-chip timer CM23 match signal	External I/O ($\overline{\text{DMAAK1}}$)	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)
			I/O ↔ Memory		I/O ↔ Memory	
			DIR = 0 or 1		Same direction as main channel	
	1	0	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)
			I/O ↔ Memory		I/O ↔ Memory	
			DIR = 0 or 1		Reverse direction as main channel	
	1	1	Parallel interface INTPAI	Data input/output buffer (PAB)	Serial reception completion INTSR0	Receive buffer (RxB0)
			I/O ↔ Memory		I/O → Memory	
			DIR = 0 or 1[Note 2]		Same direction as main channel	
Memory-to-memory transfer	0	0	Setting prohibited			
	0	1	On-chip timer CM23 match signal	–		
	1	0	DMARQ1	–		
	1	1	Setting prohibited			

Notes 1. The sub-channel is only valid in the 2-channel operating mode.

2. If the value set in DIR is 1 (memory → I/O), and the parallel interface is selected as the start source, the sub-channel side cannot be used.

6.3.4 Transfer modes

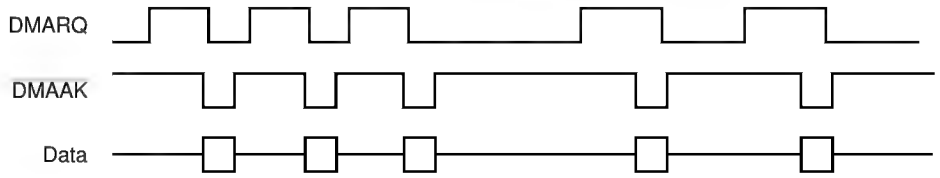
Four transfer modes can be selected for each DMA channel. How to start transfer cycles depends on the transfer mode. The transfer mode is specified with the TMD bit of the DMAC register.

(1) Single transfer mode

One DMA transfer cycle is executed each time a DMA transfer request is generated. Because the bus mastership is released immediately after the completion of the DMA transfer cycle, the CPU can shift to instruction execution again. The rising edge of the DMARQ input or the request signal from the on-chip peripheral hardware can be selected as a DMA transfer request.

Caution When DMA transfer requests are generated by several channels at a time, the DMA controller has the bus mastership until all channel requests are completed except for bus release by the hold mode and refresh cycle.

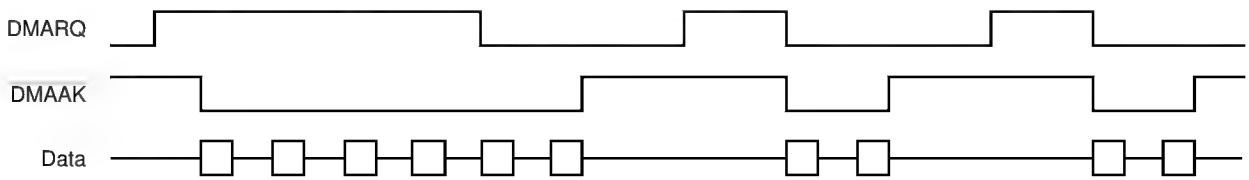
Figure 6-4. Single Transfer Mode Transfer Timing



(2) Demand release mode

DMA transfer cycles are executed consecutively while the DMARQ input is in the high-level period. During this, the CPU bus cycle is held pending, and the instruction execution stops.

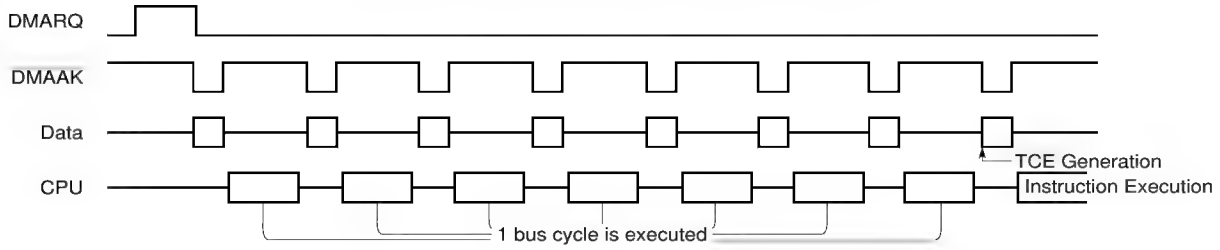
★ **Figure 6-5. Demand Release Mode Transfer Timing**



(3) Single step mode

After a DMA transfer request is generated, one DMA transfer cycle and one CPU bus cycle are executed alternately at specified times. The rising edge of the DMARQ input or the request signal from the on-chip peripheral hardware can be selected as a DMA transfer request.

Figure 6-6. Single Step Mode Transfer Timing



(4) Burst mode

After a DMA transfer request is generated, the DMA transfer cycle is executed consecutively at specified times. During this, the CPU bus cycle is held pending, and the instruction execution stops. The rising edge of the DMARQ input or the request signal from the on-chip peripheral hardware can be selected as a DMA transfer request.

Figure 6-7. Burst Mode Transfer Timing

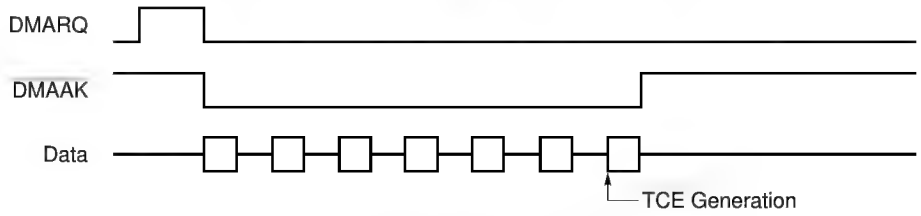


Table 6-3. Transfer Modes

Transfer Mode	DMA Start Source			Stop Method	Interrupt
	On-Chip Peripheral	Software Trigger	DMARQ Pin		
Single transfer mode	○	○	○	Reset of EDMA bit of DMAMn register	Acknowledged
Demand release mode	×	×	○	Stops when the DMARQ pin is driven low during the transfer. Reset of EDMA bit of DMAMn register	Not acknowledged during transfer. Acknowledged at other times.
Single step mode	○Note	○	○	Reset of EDMA bit of DMAMn register	Acknowledged
Burst mode	○Note	○	○	None (stop is not possible during the transfer)	Not Acknowledged

Note Transfer is possible only when the DMA start source is an on-chip timer interrupt and the transfer I/O specification is external.

6.3.5 Operating modes

There are five transfer operating modes according to the pointer control of the memory blocks of the transfer targets. The MD bit of the DMAM register is used for specification.

- Intelligent DMA mode-1
- Intelligent DMA mode-2
- Next address specification mode
- 2-channel operating mode
- Memory-to-memory mode

Table 6-4 shows the usable transfer modes and DMA interrupt (INTD) generation timings in each mode.

Table 6-4. List of Operating Modes

Operating Mode		Transfer Objects	Possible Transfer Modes[Note 1]				DPTC (IOP)	INTD Generation Conditions
			<1>	<2>	<3>	<4>		
Intelligent DMA mode-1 (ring buffer method)		I/O (SFR[Note 2]) → Memory	Yes	Yes	No	No	Usable	A
Intelligent DMA mode-2 (counter control method)		Memory → I/O (SFR[Note 2])	No	No	Yes	Yes	Usable	B
Next address specification mode		I/O (SFR[Note 2]) - Memory	Yes	Yes	No	No	No	B
2-channel operating mode	(Stop at end)	I/O (SFR[Note 2]) - Memory	Yes	Yes	Yes	Yes	No	B
	(Repetition)	I/O (SFR[Note 2]) - Memory	Yes	Yes	Yes	No	No	B
Memory-to-memory transfer[Note 3] mode	(Stop at end)	Memory (SFR) - Memory (SFR)	Yes	No	Yes	Yes	No	B
	(Repetition)	Memory (SFR) - Memory (SFR)	Yes	No	Yes	No	No	B

Notes 1. Transfer modes (See **6.3.4 Transfer modes** for details of each operation.)
<1> Single transfer mode
<2> Demand release mode
<3> Single step mode
<4> Burst mode

2. Only PAB, RxB0, RxB1, TxB0, TxB1 can be specified.

3. When executing memory-to-memory transfer for SFR, the demand release mode cannot be used, because during SFR access, no bus cycle is generated and the number of transfers cannot be controlled.

Remark INTD generation conditions

A: When the DMA up/down counter (UDC) and DMA compare register (DCM) values match as the result of executing a DMA transfer; or when the data is read by an IN or INM instruction using the input/output read/write pointer (IOP).

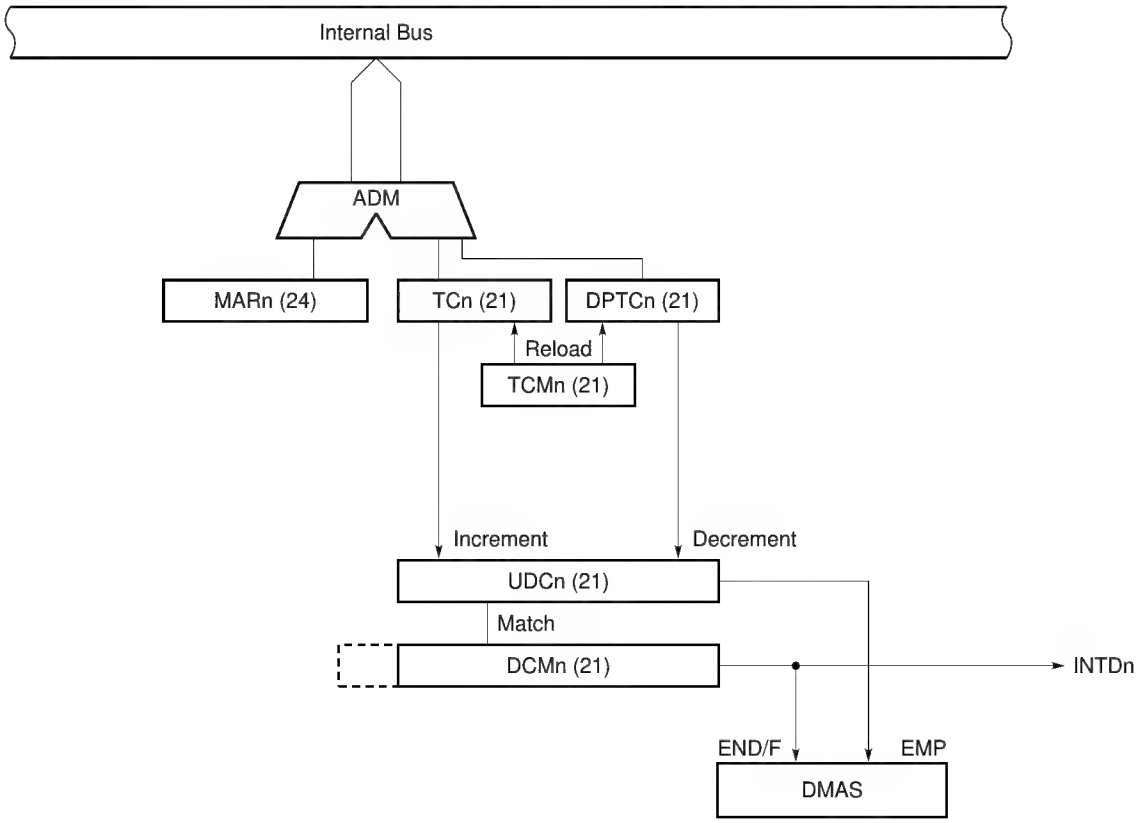
B: When a borrow is generated by a TC decrement (generation of $\overline{\text{TCE}}$ signal).

The register settings and operation in each operation mode are explained below.
For more detailed information about registers, see **6.6 Control Registers** and **6.7 DMA Service Channels**.

(1) Intelligent DMA mode-1 (ring buffer method)

In this mode the area specified as the DMA transfer destination memory has a logical ring structure (ring buffer), and repeated DMA transfers are performed from an I/O to memory.
The function block diagram for intelligent DMA mode-1 is shown in Figure 6-8.

Figure 6-8. Intelligent DMA Mode-1 (Ring Buffer Method)



[Register Settings]

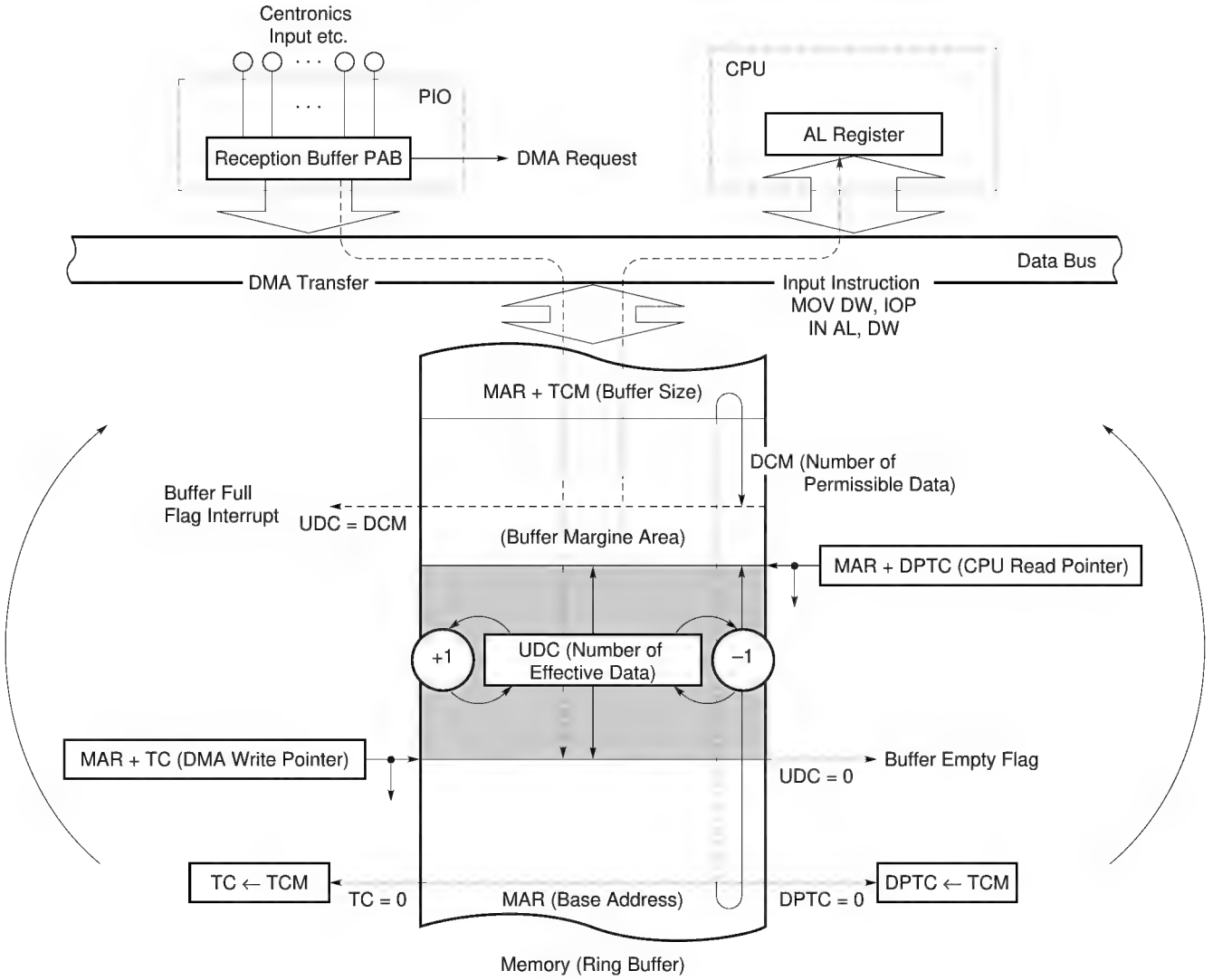
Register	Setting
MARn (DMA memory address register)	DMA transfer buffer end address (base address)
TCMn (Terminal counter modulo register) TCn (Terminal counter) DPTCn (DMA read/write pointer)	Buffer size – 1 (during byte transfer) Buffer size – 2 (during word transfer)Note 1
UDCn (DMA up/down counter)	Indicates number of valid data
DCMn (DMA compare register)	Buffer size – buffer marginNote 2
PD bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

Notes 1. For word transfer, set the buffer size with an even number.

2. The buffer margin serves to store transferred data from the time when a DMA interrupt request (INTD) is generated until the DMA transfer is stopped (EDMA is set to "0" in the DMA mode register (DMAM).

Figure 6-9 shows the buffer memory image in decrement mode.

Figure 6-9. Buffer Memory in Decrement Mode (Intelligent DMA Mode-1)



Remark The shaded area shows the area in which data is stored.

[Operation description]

(a) DMA transfer

<1> DMA request generation

<2> Data transfer from I/O to memory

Transfer destination memory addresses change as follows:

- In increment mode : MAR – TC
- In decrement mode : MARC + TC

<3> Decrement TC by –1 (for word transfer, decrement by –2).
Increment UDC by +1 (for word transfer, increment by +2).

<4> DMA transfer end

<TC borrow occurrence (if ring buffer limit is exceeded)>

The value of TCM is automatically reloaded to TC. As a result, the address becomes the ring buffer start address indicated by MAR – TC (in increment mode) or MAR + TC (in decrement mode).

(b) Read transfer data

<1> Read data from the input/output read/write pointer (IOPn) (IN instruction execution)
Execute the IN instruction using the same unit as for DMA transfer. Transfer destination memory addresses change as follows:

- Increment mode : MAR – DPTC
- Decrement mode : MAR + DPTC

<2> Decrement DPTC and UDC by –1 (for word transfer, decrement by –2).
The IOP and DPC correspondence table is shown in Table 6-5. Read in byte units when using an external 8-bit bus.

<DPTC borrow occurrence (if ring buffer limit is exceeded)

The TCM value is automatically reloaded to DPTC. As a result, the address becomes the start address of the ring buffer indicated by MAR – DPTC (in increment mode) or MAR + DPTC (in decrement mode).

Table 6-5. Relationship Between IOPn and DPTCn

Input/Output Read/Write Pointer (I/O Address)	DMA Read/Write Pointer (SFR Address)
IOP0 (FF94H)Note	DPTC0 (0FFF94H)
IOP1 (FFB4H)Note	DPTC1 (0FFFB4H)

Note It is possible to perform an access to memory using IOPn and the corresponding DPTCn. IOP0 and IOP1 function as I/O reference pointers for the data memory space (16 Mbytes).

Example Instruction sequence for reading data from memory address 23000H

```
MOV     MAR0, 2000H
MOV     MAR0H, 02H
MOV     DPTC0, 1000H
IN      AL, IOP0
```

(c) Control of number of valid data

By checking the value of UDC, it is possible to know the number of valid data that have not yet been read. The UDC is incremented by +1 (+2 for word transfer) every time one DMA transfer is executed. It is decremented by –1 (–2 for word transfer) each time data is read from IOP by executing the IN instruction.

<When UDC and DCM values match (Ring buffer is full)>

The buffer full flag (END/F) is set (to "1"). At the same time, the DMA interrupt request (INTD) is generated. If data is read from the IOP area with the IN instruction after the buffer full status, UDC is decremented by –1, the END/F flag is reset (to "0"), and at the same time a DMA interrupt request (INTD) is generated.

<When borrow is generated from UDC (no more valid data in ring buffer)>

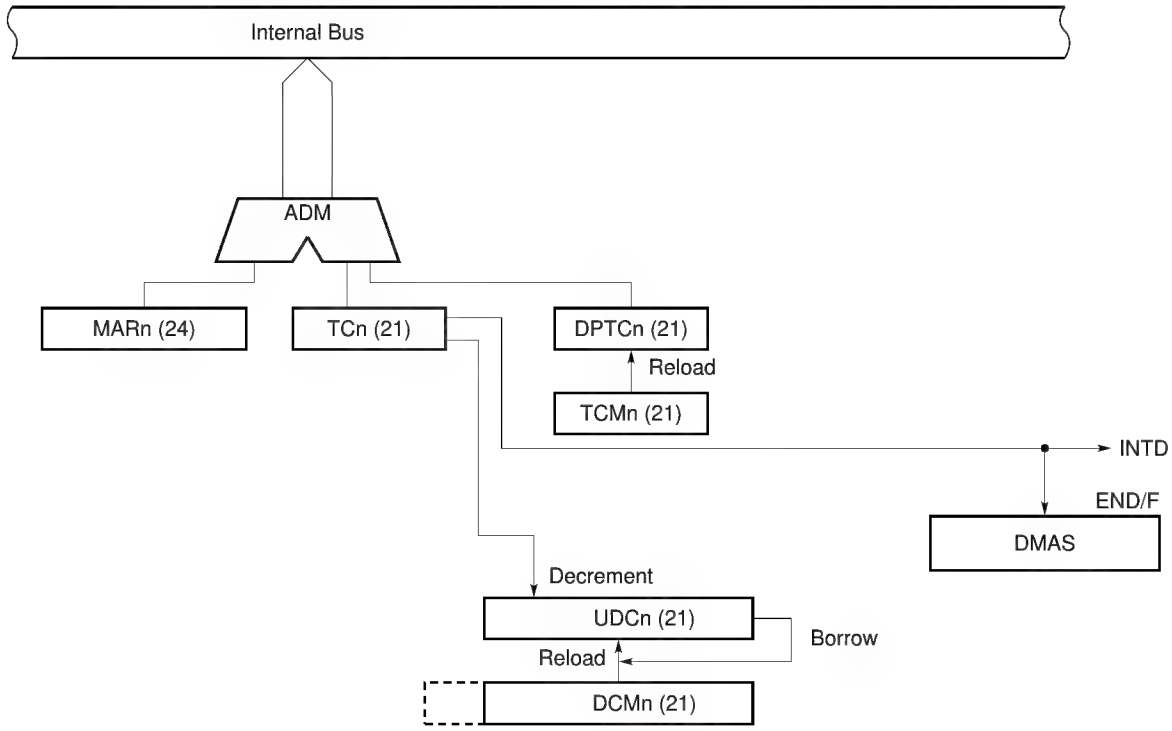
The buffer empty flag (EMP) is set (to "1"). No interrupt request (INTD) is generated.

(2) Intelligent DMA mode-2 (counter control method)

In this mode the data laid out in the area specified as the DMA transfer source is transferred by multiple DMA transfers for each DMA start request.

The function block diagram for intelligent DMA mode-2 is shown in Figure 6-10.

Figure 6-10. Intelligent DMA Mode-2 (Counter Control Method)



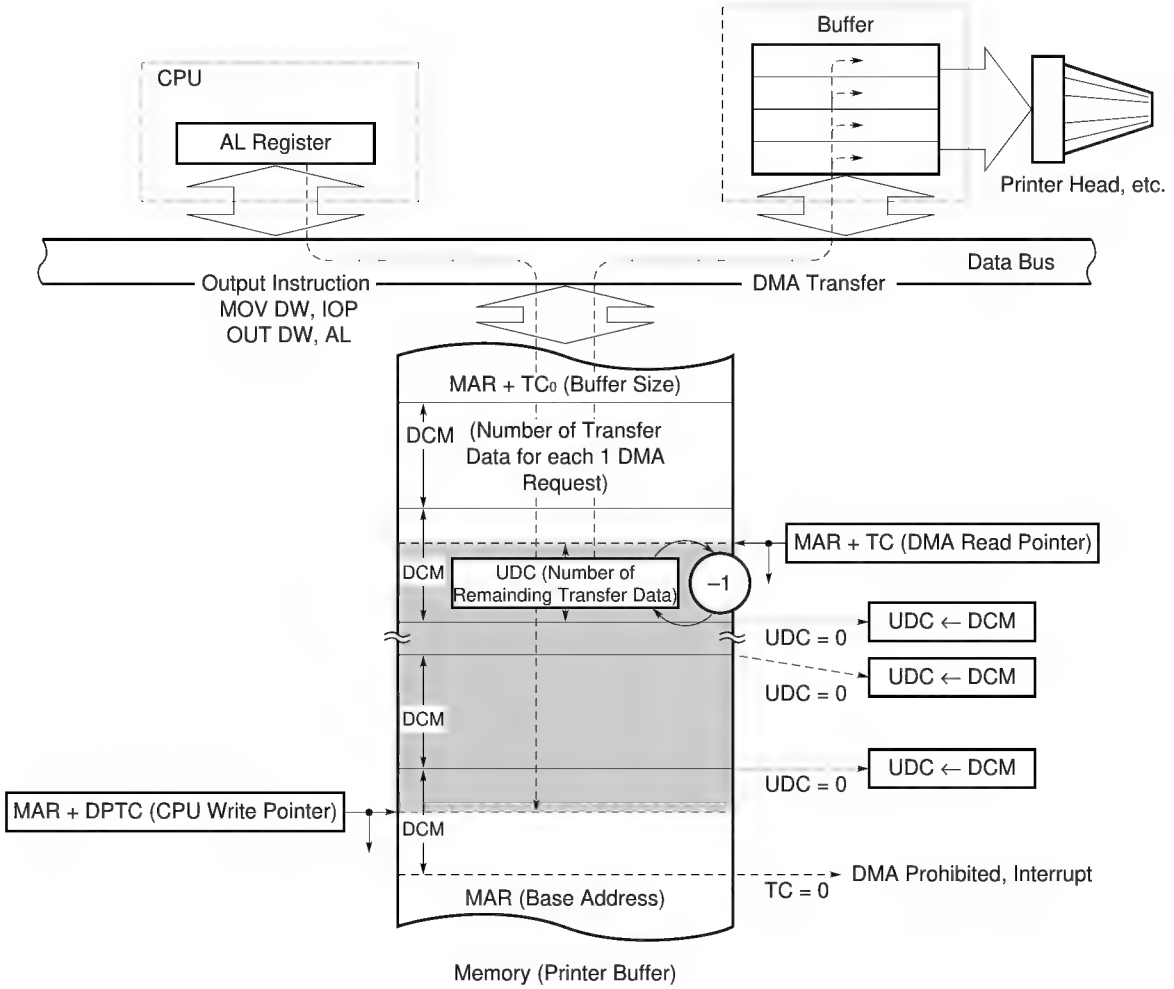
[Register setting]

Register	Description
MARn (DMA memory address register)	DMA transfer buffer memory end address (base address)
TCMn (Terminal counter modulo register) TCn (Terminal counter) DPTCn (DMA read/write pointer)	Buffer size – 1 (during byte transfer) Buffer size – 2 (during word transfer)**Note**
DCMn (DMA compare register UDCn (DMA up/down counter)	Number of DMA transfers to be executed per DMA start request – 1 (during byte transfer) Number of DMA transfers to be executed per DMA start request – 2 (during word transfer)
PD bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

Note For word transfer, set the buffer size with an even number.

Figure 6-11 shows the buffer memory image decrement mode.

Figure 6-11. Buffer Memory in Decrement Mode (Intelligent DMA Mode-2)



Remark The shaded area shows the area in which data is stored.

[Operation description]

(a) Transfer data write to transfer buffer memory

<1> Data write (OUT instruction execution) to the input/output read/write pointer (IOPn) allocated to the I/O space

Execute the OUT instruction using the same unit as the DMA transfer unit. Transfer destination memory addresses change as follows:

- In increment mode : MAR – DPTC
- In decrement mode : MAR + DPTC

<2> Decrement DPTC by –1 (for word transfer, decrement by –2).

The correspondence table between IOP and DPTC is shown in Figure 6-5. When using the external 8-bit bus, access must be made in byte units.

<DPTC borrow occurrence (if buffer limit is exceeded)>

The TCM value is automatically reloaded to DPTC. As a result, the address becomes the start address indicated by MAR – DPTC (increment mode) or MAR + DPTC (decrement mode).

(b) DMA transfer

<1> DMA request generation

<2> Data transfer from memory to I/O

Memory address containing data that will be transferred change as follows.

- In increment mode : MAR – TC
- In decrement mode : MAR + TC

<3> TC and UDC are each decremented by –1 (–2 for word transfer).

<4> Steps <2> and <3> are repeated until a borrow from UDC is generated.

<5> When a borrow from UDC is generated, DMA transfer ends, and the value of DCM (lower 21 bits) is reloaded to UDC.

<TC borrow occurrence (if buffer limit is exceeded)>

A DMA interrupt request (INTD) is generated and at the same time the END/F flag is set (to "1") (DMA disable state starts). If DMA transfer is executed after that, the TC value is reset during DMA interrupt processing (the value of TCM is not automatically reloaded to TC). At this time, regardless of the value of UDC, the DMA disable state is entered, and therefore, the value of TC must be set as follows.

Byte transfer : $(N \times (UDC + 1) - 1) \times 1$

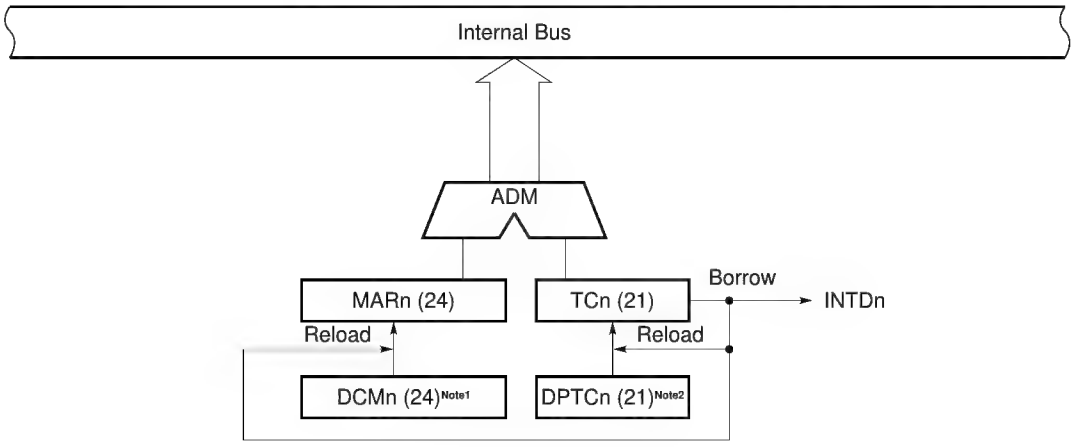
Word transfer : $(N \times (UDC + 1) - 1) \times 2$ N = integer

Caution When a DMA transfer is performed in this mode, one wait is inserted automatically even if "no wait" is specified by the PWC register. This is to allow normal borrow generation when a TC borrow is generated.

(3) Next address specification mode

In this mode, after the end of a DMA transfer in the area specified as the DMA transfer object memory, another DMA transfer is executed based on the memory address specified by the next DMA memory address register (NMARn = DCMn: DMA compare register). The DMA up/down counter (UDCn) is not used in this mode. The function block diagram in the next address specification mode is shown in Figure 6-12.

Figure 6-12. Next Address Specification Mode



Notes 1. DCMn and NMARn share the same location.

2. DPTCn and NTCn share the same location.

[Register setting]

Register	Setting
MARn (DMA memory address register)	DMA transfer end address (base address)
NMARn (next DMA memory address register)	End address of DMA transfer buffer memory to be executed next
TCn (Terminal counter)	Memory size corresponding to the difference between the start and end address – 1 (byte transfer) Memory size corresponding to the difference between the start and end address – 2 (word transfer)
DPTCn (DMA read/write pointer)Note	Memory size corresponding to the difference between the start address and end address of the DMA transfer to be executed next – 1 (byte transfer) Memory size corresponding to the difference between the start and the end address of the DMA transfer to be executed next – 2 (word transfer)
PD bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

Note In this mode, DPTC is used as NTC (next terminal counter).

[Operation description]

(a) One DMA transfer

<1> DMA request generation

<2> Data transfer between I/O and memory

Memory addresses from which data will be transferred change as follows.

- In increment mode : MAR – TC
- In decrement mode : MAR + TC

<3> Decrement TC by –1 (for word transfer, decrement by –2)

<TC borrow occurrence (if ring buffer limit is exceeded)>

The NMAR register contents are reloaded to the MAR register, and the NTC register contents are reloaded to TCNote. At the same time, an interrupt request (INTD) is generated.

At this time, the END/F flag does not change. In other words, when the next DMA request is generated, DMA transfer is continued in accordance to the newly set TC and MAR, since the DMA enable state remains unchanged.

Note When the borrow is generated, the contents of the NTC register are temporally loaded to the TCM register. Loading from the TCM register to the TC register occurs immediately before the next DMA cycle is executed.

Therefore, if the TC register is read immediately after a borrow occurs, a value is read out as it is.

(b) Read and write of transfer data from memory

Memory is usually read and written with memory manipulation instructions. No operation is performed by the input/output read/write pointer (IOPn).

Cautions 1. When a TC value is to be changed in the DMA interrupt servicing, disable the DMA transfer before changing. In this case, be sure to perform one of the following manipulations.

- **Write a TC value to be newly set into the TCM register.**
- **Write "1" into END/F flag of the DMAS register (As returned to the initial state, a TC value is counted down as it is.).**

However, if the next DMA request is generated before prohibiting a DMA transfer, the TCM value is loaded to TC to execute DMA transfer.

2. For DMA in the next address specification mode, set 1 or more in the case of byte transfer, and 2 or more in the case of word transfer for TC and DPTC (NTC).

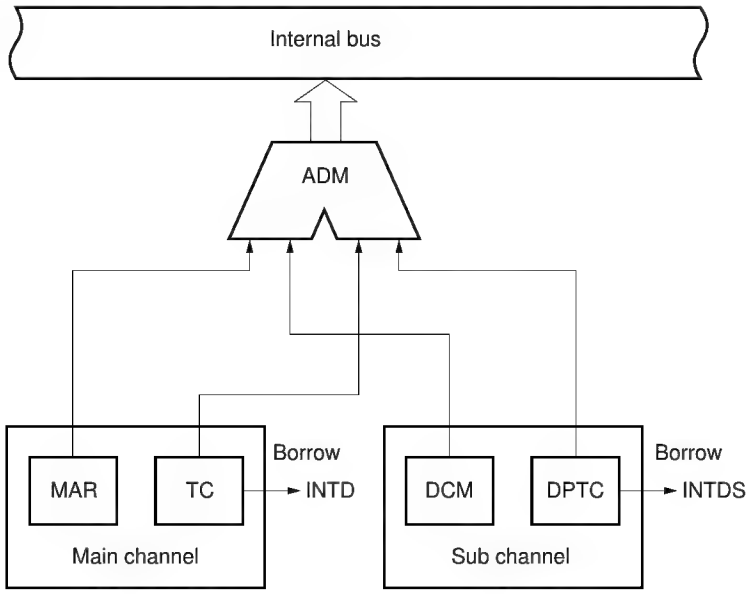
(4) 2-channel operating mode

In this mode, channel 0 and channel 1 are controlled as two channels (a main channel and a sub-channel), and DMA transfers are executed in response to the two respective DMA transfer request sources. Both the main and the sub-channels can operate in two modes: a mode in which DMA transfers end after the specified number of transfers, and a mode in which consecutive DMA transfers are performed repeatedly from the same transfer area.

The function block diagrams of each mode are shown in Figures 6-13 and 6-14.

(a) Mode in which transfer stops at end of DMA transfer (DMAMn register MD2 = 1, MD1 = MD0 = 0)

Figure 6-13. 2-Channel Operation Mode (Stop Mode)



[Register setting]

- **Main channel**

Register	Setting
MARn (DMA memory address register)	DMA transfer buffer memory end address (base address)
TCn (Terminal counter)	No. of transfer bytes – 1 (byte transfer) No. of transfer bytes – 2 (word transfer)
PD bit of DMAC (DMAC control register)	Address update direction (See **6.6.2**)

- **Sub-channel**

Register	Setting
DCMn (DMA compare register) = sub-MARn	DMA transfer buffer memory end address (base address)
DPTCn (DMA read/write pointer) = sub-TCn	No. of transfer bytes – 1 (byte transfer) No. of transfer bytes – 2 (word transfer)
PS bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

[Operation description]

<1> DMA request generation
<2> Data transfer between I/O and memory
Memory addresses from which data will be transferred change as follows.
- In increment mode : MAR – TC
- In decrement mode : MAR + TC

<3> Main channel : Decrement TC (by –2 for word transfer)
Sub-channel : Decrement sub-TC (by –2 for word transfer)

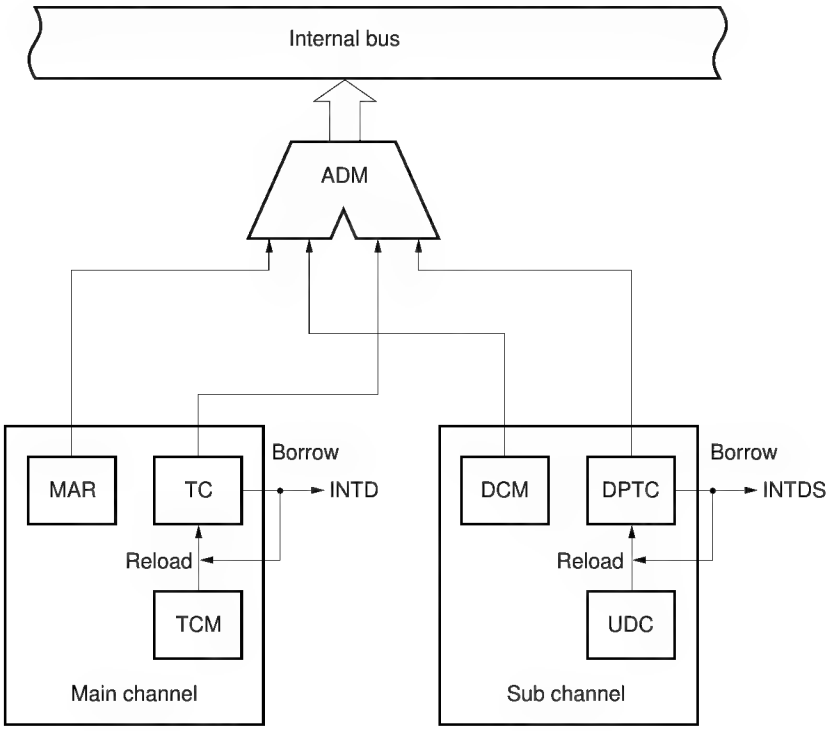
<TC (sub-TC) borrow occurrence (if buffer limit is exceeded)>

Main channel : DMA interrupt request (INTD) is generated, and at the same time the END/F flag is set (to "1"), and the DMA transfer disable state is entered.

Sub-channel : DMA interrupt request (INTDS) is generated, and at the same time the EMP flag is set (to "1"), and the DMA transfer disable state is entered.

(b) Mode in which repeated transfers are continued at end of DMA transfer (DMAMn register MD2 = MD0 = 1, MD1 = 0)

Figure 6-14. 2-Channel Operation Mode (Repeat Mode)



[Register setting]

- **Main channel**

Register	Setting
MARn (DMA memory address register)	DMA transfer buffer memory end address (base address)
TCn (Terminal counter) TCMn (Terminal counter modulo register)	No. of transfer bytes – 1 (byte transfer) No. of transfer bytes – 2 (word transfer)
PD bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

- **Sub-channel**

Register	Setting
DCMn (DMA compare register) = sub-MARn	DMA transfer buffer memory end address (base address)
DPTCn (DMA read/write pointer) = sub-TCn UDCn (DMA up/down counter) = sub-TCMn	No. of transfer bytes – 1 (byte transfer) No. of transfer bytes – 2 (word transfer)
PS bit of DMAC (DMA control register)	Address update direction (See **6.6.2**)

[Operation description]

<1> DMA request generation

<2> Data transfer between I/O and memory

Memory addresses from which data will be transferred change as follows.

- In increment mode : MAR – TC
- In decrement mode : MAR + TC

<3> Main channel : TC is decremented by –1 (–2 for word transfer)

Sub-channel : TC is decremented by –1 (–2 for word transfer)

<TC (sub-TC) borrow occurrence (if buffer limit is exceeded)>

Main channel : A DMA interrupt request (INTD) is generated, and at the same time the contents of the TCM register are automatically reloaded to TC. At this time, the END/F flag remains unchanged, and the DMA does not enter the DMA transfer disable state. Therefore, DMA transfer is repeated in the same DMA transfer area.

Sub-channel : A DMA interrupt request (INTDS) is generated, and at the same time the contents of the sub-TCM are automatically reloaded to TC. At this time, the EMP flag remains unchanged, and the DMA does not enter the DMA transfer disable state. Therefore, DMA transfer is repeated in the same DMA transfer area.

Remark A currently executed DMA transfer is stopped when the DMA enable flag (EDMA) is cleared (to "0") by software, for example DMA interrupt processing.

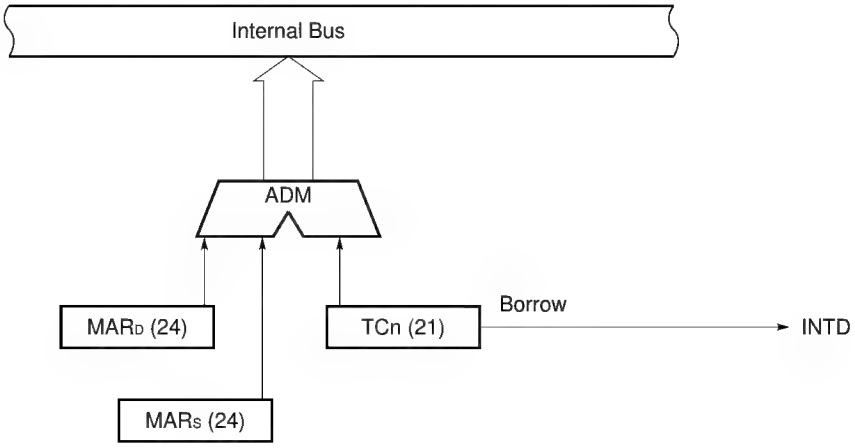
(5) Memory-to-memory transfer mode

In this mode, two bus cycles are started consecutively and DMA transfer is executed between memory locations. In this mode the DMA up/down counter (UDCn) and DMA read/write pointer (DPTCn) are not used. There are in fact two memory-to-memory transfer modes: a mode in which DMA transfer stops after the specified number of DMA transfers, and a mode in which consecutive DMA transfers are performed repeatedly from the same transfer areas.

Figures 6-15 and 6-16 show the function block diagram in each mode.

(a) Mode in which transfer stops at end of DMA transfer (DMAMn register MD2 = MD1 = 1, MD0 = 0)

Figure 6-15. Memory-to-Memory Transfer Mode (Stop Mode)



[Register setting]

Register	Description
DCMn (DMA compare register) = MAR_S	Source side DMA transfer buffer memory end address
MARn (DMA memory address register) = MAR_D	Destination side DMA transfer buffer memory end address
TCn (Terminal counter)	No. of transfer bytes – 1 (for byte transfer) No. of transfer bytes – 2 (for word transfer)
PS bit of DMAC (DMA control register)	Source side address transfer direction (see **6.6.2**)
PD bit of DMAC (DMA controller register)	Destination side address transfer direction (See **6.6.2**)

[Operation description]

<1> DMA request generation

<2> Inter-memory data transfer

Memory addresses from which data will be transferred are as follows.

	Source side	Destination side
Increment mode	MAR_S – TC	MAR_D – TC
Decrement mode	MAR_S + TC	MAR_D + TC

However, when the address has not been updated, only MAR_S (source side) or MAR_D (destination side) is specified.

<3> TC is decremented by –1 (–2 for word transfer)

<TC borrow occurrence (if buffer limit is exceeded)>

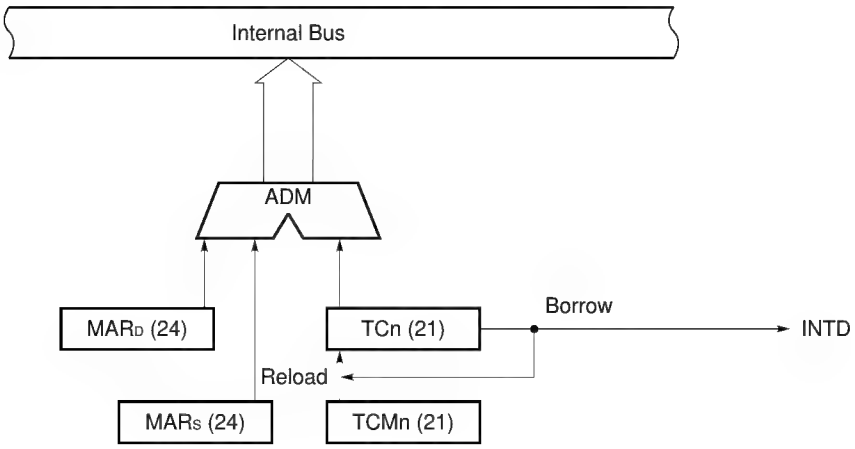
A DMA interrupt request (INTD) is generated, and at the same time the END/F flag is set (to "1"), and the DMA transfer disable state is entered. Also, the contents of TCM are reloaded to TC[Note].

Note The contents of the TCM register are loaded to the TC register immediately before the next DMA cycle is executed.

Therefore, if the TC register is read immediately after a borrow occurs, a value is read out as it is.

(b) Mode in which repeated transfer are continued at the end of DMA transfer (DMAMn register MD2 = MD0 = MD1 = 1)

Figure 6-16. Memory-to-Memory Transfer Mode (Repeat Mode)



[Register setting]

Register	Description
DCMn (DMA compare register) = MARS	Source side DMA transfer buffer memory end address
MARn (DMA memory address register) = MARD	Destination side DMA transfer buffer memory end address
TCn (Terminal counter) TCMn (Terminal counter modulo register)	No. of transfer bytes – 1 (for byte transfer) No. of transfer bytes – 2 (for word transfer)
PS bit of DMAC (DMA control register)	Source side address transfer direction (see **6.6.2**)
PD bit of DMAC (DMA control register)	Destination side address transfer direction (See **6.6.2**)

[Operation description]

<1> DMA request generation

<2> Inter-memory data transfer

Memory addresses from which data will be transferred are as follows.

	Source side	Destination side
Increment mode	MAR_S – TC	MAR_D – TC
Decrement mode	MAR_S + TC	MAR_D + TC

However, when the address has not been updated, only MAR_S (source side) or MAR_D (destination side) is specified.

<3> TC is decremented by –1 (–2 for word transfer)

<TC borrow occurrence (if buffer limit is exceeded)>

A DMA interrupt request (INTD) is generated, and at the same time the contents of TCM are automatically reloaded to TC[Note]. At this time, the END/F flag remains unchanged, and DMA transfer is repeated in the same DMA transfer area.

Note The contents of the TCM register are loaded to the TC register immediately before the next DMA cycle is executed.
Therefore, if the TC register is read immediately after a borrow occurs, a value is read out as it is.

Remark DMA transfer is stopped if the DMA enable flag (EDMA) is cleared (to 0) by software, for example by means of DMA interrupt processing.

6.4 Operation

A series of DMA transfers is executed in several DMA transfer cycles. The start, stop, and suspend of DMA transfer cycle are described below.

6.4.1 Start

In the DMA transfer enable state (EDMA and EDMAS bits of the DMAM register are "1"), the DMA transfer cycle is started with one of the following methods. The execution is in accordance with the DMA service channel settings.

(1) DMA request from a start source

The following start sources can be selected. Each start source is specified with the TS0 and TS1 bits of the DMA mode register (DMAMn) (See **6.6.1 DMA mode registers (DMAM0, DMAM1)**).

(a) Interrupt request by on-chip peripheral hardware

- Serial transmit interrupt (INTST0, INTST1)
- Serial receive interrupt (INTSR0, INTSR1)
- On-chip timer interrupt (timer 2: CM23 match signal)
- Parallel interface input/output interrupt (INTPAI)

(b) External pin input (DMARQ0, DMARQ1)

(2) DMA request by software

DMA can be started by setting the TDMA bit of the DMA mode register (DMAMn) to "1" with software. However, this method cannot be used in the demand release mode.

In the burst mode, single step mode, and demand release mode, the DMA transfer cycle continues until the stop condition is satisfied.

When a DMA request is generated during the HALT mode, the HALT mode is cleared and the DMA transfer is executed. After the DMA transfer is executed, the HALT mode is restored. In this case, the contents of the program counter is not changed.

When a DMA request is generated from outside during the STOP mode, the DMA controller does not execute the DMA transfer. The request generation is also not memorized.

6.4.2 Stop

A series of DMA transfers is stopped by one of the following stop conditions.

A DMA transfer cycle being executed is performed to the end properly. Even if the DMA transfer conflicts with the STOP or HALT instruction, a stand-by mode is not set during the DMA cycle execution. After the completion of the DMA transfer, the stand-by mode is set and the execution is stopped.

(1) Conclusion of DMA transfer termination condition

- When the EMP flag of the DMA status register (DMAS) is set (1) in the 2-channel operating mode:
- When the END/F flag of the DMA status register (DMAS) is set (1):

(2) Manipulation of EDMA and EDMAS bits.

- When the EDMA bit of the DMA mode register (DMAMn) is cleared (0):
- When the EDMAS bit of the DMA mode register (DMAMn) is cleared (0) (Only the sub-channel DMA operation in the 2-channel operating mode is stopped):

(3) Detection the low level of DMARQ input

DMA transfer is stopped only in the demand release mode.

6.4.3 Suspend

DMA transfer is suspended by the following elements.

(1) Bus hold
(2) Refresh cycle

When elements that cause the DMA transfer to be suspended are eliminated, the DMA transfer starts again. Even if another DMA request with higher-priority is generated during a DMA transfer, the DMA transfer being executing is not suspended.

6.5 Priority

6.5.1 Inter-channel priority

When DMA transfer requests are generated for several channels simultaneously, inter-channel priority is determined as follows.

(High) Channel 0 (Main channel) > Channel 0 (Sub-channel) > Channel 1 (Main channel) > Channel 1 (Sub-channel) (Low)

Caution There is no nesting concept in inter-channel priority control; once a DMA request is acknowledged, transfer continues until a DMA transfer end condition arises. Care is therefore required when using the burst mode.

6.5.2 Priority with respect to refresh cycle

If a refresh request and a DMA request are generated simultaneously, the refresh cycle has priority and the DMA transfer is held pending.

6.5.3 Priority with respect to other control operations

When DMA control register manipulation is performed by an instruction, the DMA execution disabled state (requests pending) is automatically set for both channels.

The following conditions also need to be considered in addition to contention with a normal CPU data access, instruction fetch, etc., through bus monopolization by DMA.

<1> Memory refresh operation
<2> Standby operation
<3> NMI input
<4> Watchdog timer overflow interrupt (INTWDT)
<5> External hold request

The relative priorities of these conditions and DMA is as follows:

External hold request > memory refresh operation > DMA > standby operation

The operation of the bus control unit has no effect with regard to contention between <3> or <4> and DMA. That is, because the DMA controller has bus mastership during a DMA operation, DMA has priority even if a bus request is generated in an NMI or INTWDT service program (processing in which a bus request is not generated is executed in parallel).

In the case of contention between an access to external memory by the CPU and a DMA or refresh operation, the start of the bus cycle is held pending for whichever requested as access later. However, a memory access and a DMA request are generated simultaneously, the memory access has priority.

6.6. Control Registers

DMA mode registers (DMAMn) and DMA control registers (DMACn) are provided to perform DMA operating mode and transfer mode specification. The DMA status register (DMAS) is provided to indicate the DMA transfer end status, etc.

6.6.1 DMA mode registers (DMAM0, DMAM1)

These are 8-bit registers used to specify the DMA operating mode, DMA transfer enabling/disabling, etc.
They can be written to or read by an 8-bit or 1-bit memory manipulation instruction.
Their value after a reset is E0H. Hereafter, shaded area in a table indicates status after a reset.

DMAM0
DMAM1

7	6	5	4	3	2	1	0
MD2	MD1	MD0	EDMAS	EDMA	TDMA	TS1	TS0

MD2, MD1, MD0 Operating mode specification

Specify the operating mode.

MD2	MD1	MD0	Operating Mode	
0	0	0	I/O (SFR) → memory transfer	Intelligent DMA mode-1 (ring buffer method)
0	0	1	I/O (SFR) ← memory transfer	Intelligent DMA mode-2 (counter control method)
0	1	0	I/O (SFR) ↔ memory transfer	Next address specification mode
0	1	1	Setting prohibited	
1	0	0	I/O ↔ memory transfer	2-channel operating mode (stop at end)
1	0	1	I/O ↔ memory transfer	2-channel operating mode (repetition)
1	1	0	Memory ↔ memory transfer	Memory-to-memory transfer mode (stop at end)
1	1	1	Memory ↔memory transfer	Memory-to-memory transfer mode (repetition)

EDMAS DMA transfer enable/disable control bit (sub-channel)

Specifies enabling/disabling of sub-channel DMA transfers in the 2-channel operating mode.

EDMAS	DMA Transfer Enable/Disable
0	DMA transfer disabled state
1	DMA transfer enabled state

When the EMP flag in the DMA status register (DMAS) is set (1), setting (1) the EDMAS bit does not set the DMA transfer enabled state.

EDMA DMA transfer enable/disable control bit

Specifies enabling/disabling of DMA transfers.

EDMA	DMA Transfer Enable/Disable
0	DMA transfer disabled state
1	DMA transfer enabled state

When the END/F flag in the DMA status register (DMAS) is set (1), setting (1) the EDMA bit does not set the DMA transfer enabled state.

TDMA DMA transfer software start bit

Trigger bit used to start a DMA transfer by software.

EDMA	TDMA	Transfer State
0	×	DMA transfer disabled state
1	0	State maintained
1	1	DMA transfer start (software trigger)

A DMA transfer cannot be started when the END/F flag in the DMA status register (DMAS) is set (1).
The read level of this bit is always (0).
A transfer cannot be started by a software trigger in the demand release mode.
In the 2-channel operating mode, starting a transfer by a software trigger is possible only for the main channel in accordance with the TDMA bit.

TS1, TS0 DMA start request and transfer I/O specification

A DMA start request can be selected from a maximum of 4 sources by means of the TS1 and TS0 bits.
In the following table, the DMA request specified by the main channel and sub channel is selected in the 2-channel operating mode, and the DMA request specified by the main channel is selected in modes other than the 2-channel operating mode.

(Channel 0)

Corresponding Transfer Mode (Transfer Object)	TS1	TS0	Main Channel		Sub-Channel	
			DMA Start Trigger	Transfer I/O	DMA Start Trigger	Transfer I/O
I/O-to-memory transfer	0	0	Parallel interface INTPAI	Data input/output buffer (PAB)	Serial transmission completion INTST0	Transmit buffer (TxB0)
			Transfer direction (I/O ↔ Memory[Note])		Transfer direction (Memory → I/O)	
	0	1	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)
			Transfer direction (I/O ↔ Memory)		Transfer direction (Same direction as main channel)	
	1	0	On-chip timer CM23 match signal	External I/O ($\overline{\text{DMAAK0}}$)	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)
			Transfer direction (I/O ↔ Memory)		Transfer direction (Reverse direction to main channel)	
	1	1	Serial transmission completion INTST0	Transmit buffer (TxB0)	Serial transmission completion INTST1	Transmit buffer (TxB1)
			Transfer direction (Memory → I/O)		Transfer direction (Memory → I/O)	
Memory-to-memory transfer	0	0	Setting prohibited			
	0	1	DMARQ0	–		
	1	0	CM23 match signal	–		
	1	1	Setting prohibited			

Note If memory → I/O (DIR = 1) is selected as the main channel transfer direction, the sub-channel side cannot be used.

Caution In a memory-to-memory transfer, $\overline{\text{DMAAKn}}$ and the acknowledge signal to the on-chip timer and serial control circuit are not output.

(Channel 1)

Corresponding Transfer Mode (Transfer Object)	TS1	TS0	Main Channel		Sub-Channel	
			DMA Start Trigger	Transfer I/O	DMA Start Trigger	Transfer I/O
I/O-to-memory transfer	0	0	Serial reception completion INTSR1	Receive buffer (RxB1)	Serial reception completion INTST0	Transmit buffer (TxB0)
			Transfer direction (I/O → Memory)		Transfer direction (Memory → I/O)	
	0	1	On-chip timer CM23 match signal	External I/O ($\overline{\text{DMAAK1}}$)	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)
			Transfer direction (I/O ↔ Memory)		Transfer direction (Same direction as main channel)	
	1	0	DMARQ1	External I/O ($\overline{\text{DMAAK1}}$)	DMARQ0	External I/O ($\overline{\text{DMAAK0}}$)
			Transfer direction (I/O ↔ Memory)		Transfer direction (Reserve direction to main channel)	
	1	1	Parallel interface INTPAI	Data input/output buffer (PAB)	Serial reception completion INTSR0	Receive buffer (RxB0)
			Transfer direction (I/O ↔ Memory)[Note]		Transfer direction (I/O → Memory)	
Memory-to-memory transfer	0	0	Setting prohibited			
	0	1	CM23 match signal	–		
	1	0	DMARQ1	–		
	1	1	Setting prohibited			

Note If memory → I/O (DIR = 1) is selected as the main channel transfer direction, the sub-channel side cannot be used.

Caution In a memory-to-memory transfer, $\overline{\text{DMAAKn}}$ and the acknowledge signal to the on-chip timer and serial control circuit are not output.

6.6.2 DMA control registers (DMAC0, DMAC1)

These are 8-bit registers which specify the DMA transfer mode, memory source address and destination address update method, etc. These registers can be read/written to by an 8-bit or 1-bit memory manipulation instruction.

Their value after a reset is 00H.

DMAC0, DMAC1

7	6	5	4	3	2	1	0
TMD1	TMD0	DIR	W	PD1	PD0	PS1	PS0

Address Update Specification

PD1/PS1	PD0/PS0	Address Update Mode
0	0	Non-Update
0	1	Increment
1	0	Decrement
1	1	Non-Update

Transfer Processing Method Specification

W	Transfer Processing Method
0	Byte transfer
1	Word transfer

Transfer Direction Specification

DIR	Transfer Direction
0	I/O → Memory transfer
1	Memory → I/O transfer

Transfer Mode Specification

TMD1	TMD0	Transfer Mode
0	0	Single transfer mode
0	1	Demand release mode
1	0	Burst mode
1	1	Single step mode

Remark The shaded area indicates status after a reset.

DIR Transfer direction specification bit

This bit is only valid when the intelligent DMA mode, next address specification mode, or 2-channel operating mode is specified.

In intelligent DMA mode-1, only I/O-to-memory transfers are possible, and therefore the DIR bit should be cleared (0).

Similarly, in intelligent DMA mode-2, only memory-to-I/O transfers are possible, and therefore the DIR bit should be set (1).

In memory-to-memory transfers the DIR bit must be set to "0".

PD1, PD0/PS1, PS0 Address update specification bits

The PD1 and PD0 specification determines memory address generation in each operating mode as shown below.

Type of transfer : I/O ↔ memory

Operating modes : Intelligent DMA mode-1, Intelligent DMA mode-2, next address specification mode, 2-channel operating mode

PD1	PD0	Address Update Mode	Memory Address Generation Method
0	0	Setting prohibited	–
0	1	Increment	MAR – TC
1	0	Decrement	MAR + TC
1	1	Setting prohibited	–

Type of transfer : Memory ↔ memory

Operating modes : Memory-to-memory transfer mode

PD1	PD0	Address Update Mode	Memory Address Generation Method
0	0	Non-update	MAR
0	1	Increment	MAR – TC
1	0	Decrement	MAR + TC
1	1	Non-update	MAR

The address non-update mode can only be used in the memory-to-memory transfer mode. In other operating modes, setting is prohibited.

It is also possible to perform a DM transfer on a memory-mapped I/O (such as an FIFO) rather than on normal memory using the address non-update mode.

In the address non-update mode, also, TC subtracting is performed and DMA end signal output and a DMA interrupt request (INTD) are generated.

The PS1 and PS0 bit specification determines memory address generation in each operating mode as shown below.

Type of transfer : Memory ↔ memory
Operating modes : Memory-to-memory transfer mode

PS1	PS0	Address Update Mode	Memory Address Generation Method
0	0	Non-update	DCM
0	1	Increment	DCM – TC
1	0	Decrement	DCM + TC
1	1	Non-update	DCM

Type of transfer : Memory ↔ I/O
Operating modes : Intelligent DMA mode-1, Intelligent DMA mode-2

PS1	PS0	Address Update Mode	Memory Address Generation Method
0	0	Setting prohibited	–
0	1	Increment	MAR – DPTC
1	0	Decrement	MAR + DPTC
1	1	Setting prohibited	–

The DPTC is decremented for each read/write instruction executed by one IOPn.
In the next address specification mode, PS1 and PS0 should be set to 0.

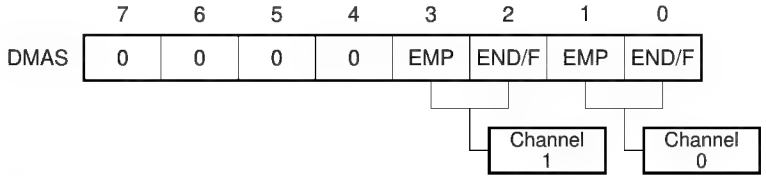
6.6.3 DMA status register (DMAS)

This is a readable/writable 8-bit register which indicates the DMA transfer end status, and the buffer full and buffer empty status for data in the ring buffer in intelligent DMA mode-1.

This register can be read/written to (reset only) by an 8-bit memory manipulation instruction. When rewriting the register contents, "0" must be set to bits 7 to 4.

Cautions 1. The register contents are not initialized even if the DMA transfer mode is switched and therefore initialization should be performed after mode switching if necessary.

2. A 1-bit manipulation instruction cannot be used. Write data in which only bits to be reset is set to "1" by the 8-bit manipulation instruction. Note that a bit test instruction can be used.



END/F Transfer end/buffer full flag

This flag is valid in intelligent DMA mode-1, intelligent DMA mode-2, 2-channel operating mode (stop at end mode), and memory-to-memory transfer mode (stop at end mode).

This flag is not changed in the next address specification mode, 2-channel operating mode (repetition), and memory-to-memory transfer mode (repetition). When this flag is set (1), DMA transfer is prohibited even if the EDMA bit of the DMAM register is set (1).

(1) Intelligent DMA mode-1

END/F functions as a flag to detect the ring buffer data full condition.

END/F is set (1) when the DMA up/down counter (UDC) is incremented by a DMA transfer and it matches the DMA compare register (DCM) set value.

END/F is reset (0) if the data in the buffer is read by an input/output instruction and the UDC is decremented when the UDC and DCM values match. This flag is also reset (0) by an operation which writes "1" by means of an instruction.

(2) Intelligent DMA mode-2, 2-channel operating mode (stop at end), memory-to-memory transfer mode (stop at end)

END/F functions as a flag which indicates completion of the number of DMA transfers set in the terminal counter (TC).

END/F is set (1) if a borrow is generated when the TC is decremented.
END/F is reset (0) by an operation which writes "1" by means of an instruction.

EMP Buffer empty flag

This flag is valid in intelligent DMA mode-1 and 2-channel operating mode. This flag is not changed in the next address specification mode, 2-channel operating mode (repetition), and memory-to-memory transfer mode (repetition). When this flag is set (1), DMA transfer is prohibited even if the EDMAS bit of the DMAM register is set (1).

(1) Intelligent DMA mode-1

EMP functions as a flag to detect the ring buffer data empty full condition.
EMP is set (1) if a borrow is generated out of the UDC when the UDC is decremented by reading data in the buffer via the input/output read/write pointer. EMP is reset (0) when a DMA transfer is executed and valid data is stored in the ring buffer. This flag is also reset (0) by an operation which writes "1" by means of an instruction.

(2) 2-channel operating mode

EMP functions as a flag which indicates completion of the number of DMA transfers set in the sub-channel terminal counter (UDC).
EMP is set (1) if a borrow is generated when the UDC is decremented, and is reset (0) by an operation which writes "1" by means of an instruction.

6.7 DMA Service Channels

The registers shown below are provided for each channel in the special function register area to specify the transfer source, transfer destination, number of transfer bytes, etc., used in a DMA transfer (see **Figure 6-3 Channel Configuration** for the channel configuration).

The registers used in each operating mode are shown in Table 6-6.

The low-order 16 bits (bits 15 to 0) of each register can only be read as a word unit by an instruction: byte access cannot be used.

Table 6-6. Registers in Each Operating Mode

Register Name (Number of Bits)	Operating Mode				
	Intelligent DMA Mode-1	Intelligent DMA Mode-2	Next Address Specification Mode	2-Channel Operating Mode	Memory-to Memory Mode
TC (21 bits)	TC	TC	TC	Main TC	TC
TCM (21 bits)	TCM	TCM	Not used	Main TCM	TCM
UDC (21 bits)	UDC	UDC	Not used	Sub-TCM	Not used
DPTC (21 bits)	DPTC	DPTC	NTC	Sub-TC	Not used
MAR (24 bits)	MAR	MAR	MAR	Main MAR	MAR_D (Destination)
DCM (24 bits)	DCM	DCM	NMAR	Sub-MAR	MAR_S (Source)

6.7.1 Terminal counters (TC0, TC1)

TC0 and TC1 are 21-bit registers which control the number of DMA transfers by setting the number of DMA transfer bytes.

The register contents are decremented each time a DMA transfer is executed, and DMA transfers end when a borrow is generated. The number of transfers should be set as follows, according to whether byte or word transfers are used:

Byte transfers : (Number of transfers – 1) × 1
Word transfers : (Number of transfers – 1) × 2

The maximum number of transfer bytes is 2,097,152 and the maximum number of word transfers is 1,048,576.

6.7.2 Terminal counter modulo registers (TCM0, TCM1)

TCM0 and TCM1 are 21-bit modulo registers which reload information on the number of DMA transfers into TCn. This information is also reloaded into the DPTC in intelligent DMA mode-1 and intelligent DMA mode-2.

6.7.3 DMA up/down counters (UDC0, UDC1)

These are 21-bit registers used in the intelligent DMA mode. In intelligent DMA mode-1 they indicate the difference between the number of DMA transfer bytes (words) and the number of CPU memory read/write bytes (words). They are incremented when one DMA transfer is executed, and decremented when a read or write is performed on the input/output read/write pointer (IOP) by an I/O manipulation instruction.

In intelligent DMA mode-2, the number of DMA transfers to be executed by one DMA start request should be set.

In the 2-channel operating mode, they are used as the sub-channel TCM registers.

6.7.4 DMA read/write pointers (DPTC0, DPTC1)

DPTCn are 21-bit registers used in the intelligent DMA mode to indicate the CPU memory read/write pointer. The read/write address (MARn ± DPTCn) is indicated by the input/output read/write pointer (IOP).

They are also used as the next terminal counters (NTCn) in the next address specification mode.

In the 2-channel operating mode, they are used as the sub-channel TC.

6.7.5 DMA memory address registers (MAR0, MAR1)

These are 24-bit registers used to indicate the memory address subject to a DMA transfer.

They specify the DMA transfer end memory address. The memory address subject to the DMA transfer is indicated by MARn ± TCn.

In the 2-channel operating mode, they are used as the main channel memory address registers.

In memory-to-memory transfers, these registers function as memory address registers (destination: MAR_{Dn}).

6.7.6 DMA compare registers (DCM0, DCM1)

DCMn registers are 24-bit registers used in the intelligent DMA mode.

In intelligent DMA mode-1, they are used to determine the buffer full condition by comparison with UDCn.

In intelligent DMA mode-2, they function as registers which hold the initial value of the number of DMA transfers reloaded into the UDC when a borrow is generated as the result of decrementing the UDC.

In the 2-channel operating mode, they are used as the sub-channel MAR registers.

In the next address specification mode, they are used as the next memory address registers (NMARn) which set the next transfer destination (source) memory address to be reloaded into the MAR register when the terminal count ends (when a borrow is generated as the result of decrementing the TC), and in the memory-to-memory transfer mode, they are used as the DMA memory address registers (source: MARsn).

6.8 Interrupt Requests

An interrupt request signal is sent to the interrupt controller from each channel of the DMA controller. The correspondence between the DMA service channels and the DMA interrupt request signal is shown below.

[Modes other than 2-channel operating mode]

DMA Channel	Interrupt Request Signal
Channel 0	INTD0
Channel 1	INTD1

[2-channel operating mode]

DMA Channel	Interrupt Request Signal
Channel 0 (main channel)	INTD0
Channel 1 (main channel)	INTD1
Channel 0 (sub-channel)	INTDS0
Channel 1 (sub-channel)	INTDS1

(1) In intelligent DMA mode-1

A DMA interrupt is generated if UDCn matches the DCMn register value (if the END/F flag is set (1)) when it is incremented, or if the data in the buffer is read by an input/output instruction via the input/output read/write pointer (IOP) and the UDC is decremented (the END/F flag is reset (0)) when the UDC and DCM register values match.

(2) In intelligent DMA mode-2, next address specification mode, 2-channel operating mode, and memory-to-memory transfer mode

An interrupt request is generated if a borrow is generated by decrementing TCn for each DMA transfer.

6.9 Response Time to DMA Request

The response time to DMA request can be obtained from the expression below.

From 6 to (9 + W + α) (Unit: clock)

W : Number of bus wait cycles immediately before a DMA transfer cycle
α : Number of delay clocks caused by conflict with bus cycle in instruction executions.

The number of clocks differs depending on the instructions as shown below.

Instructions	Number of clocks
The following instructions to SFR (destination) Logical operation, add/subtract, complement operation, bit manipulate, XCH instruction	11 + 2a + b
The following instructions to SFR (destination) Shift, rotate instruction	14 + 2a + b + n
The following instruction to SFR (destination) Compare instruction	4 + a
INS instruction	24 + 2a + b + 1 + m
EXT instruction	15 + a + b
BTCLR, BTCLRL instruction	14 + a + b
Other instructions	0

a : Manipulation to TM register: 8
Manipulation to SFR (TxBRG, RxBRG, IMC, ISPR, MK, IC, TMC, TOC, INTM, CT CM): 2
Manipulation to SFR (other than the above): 0
b : Wait number to memory
l : reg8 setting value
m : reg8' setting value
n : Number of shift times or rotate times

6.10 DMA Transfer Timing

The timing charts for the DMA transfer mode are shown below. In a DMA transfer by the DMA controller, a wait operation by means of the READY pin and the PWC0 and PWC1 registers is performed in the same way as for an ordinary bus cycle.

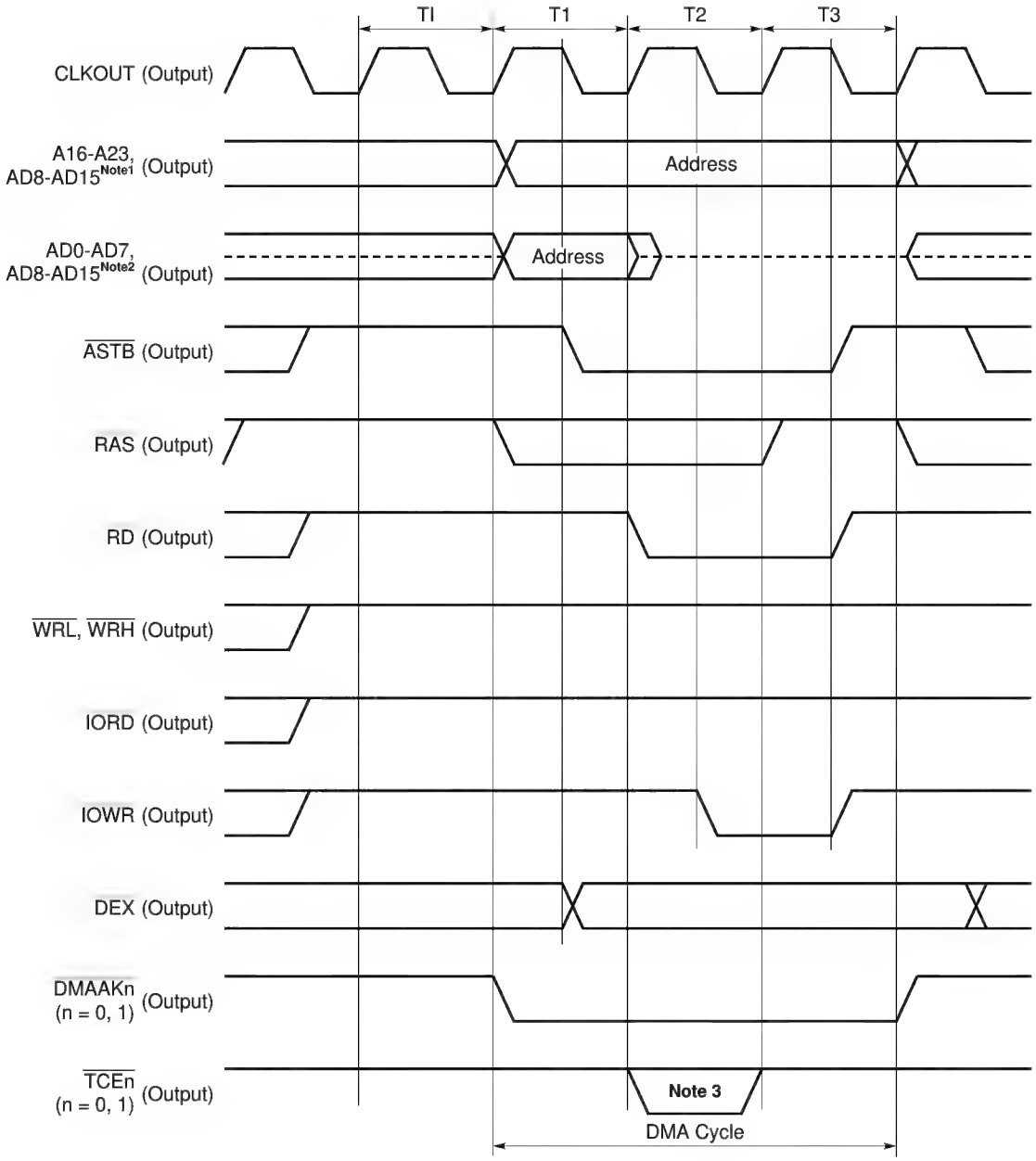
The timing charts for each transfer target are shown below.

Transfer Object	Modes	Fig. No.	Transfer Object	Modes	Fig. No.
Memory → External I/O	ST, SS	6-17	Memory ↔ Memory	ST, SS	6-21
	DR, BS			BS	
External I/O → Memory	ST, SS	6-18	Memory → SFR (Memory Access)	ST, SS	6-22
	DR, BS			BS	
Memory → Internal I/O (SFR)	ST	6-19	SFR (Memory Access) → Memory	ST, SS	6-23
				BS	
Internal I/O (SFR) → Memory	ST	6-20			

Remark ST : Single transfer mode
SS : Single step mode
DR : Demand release mode
BS : Burst mode

Figure 6-17. Memory → External I/O (1/2)

(a) Single Transfer Mode and Single Step Mode



Notes 1. When the external data bus width is 8-bit:

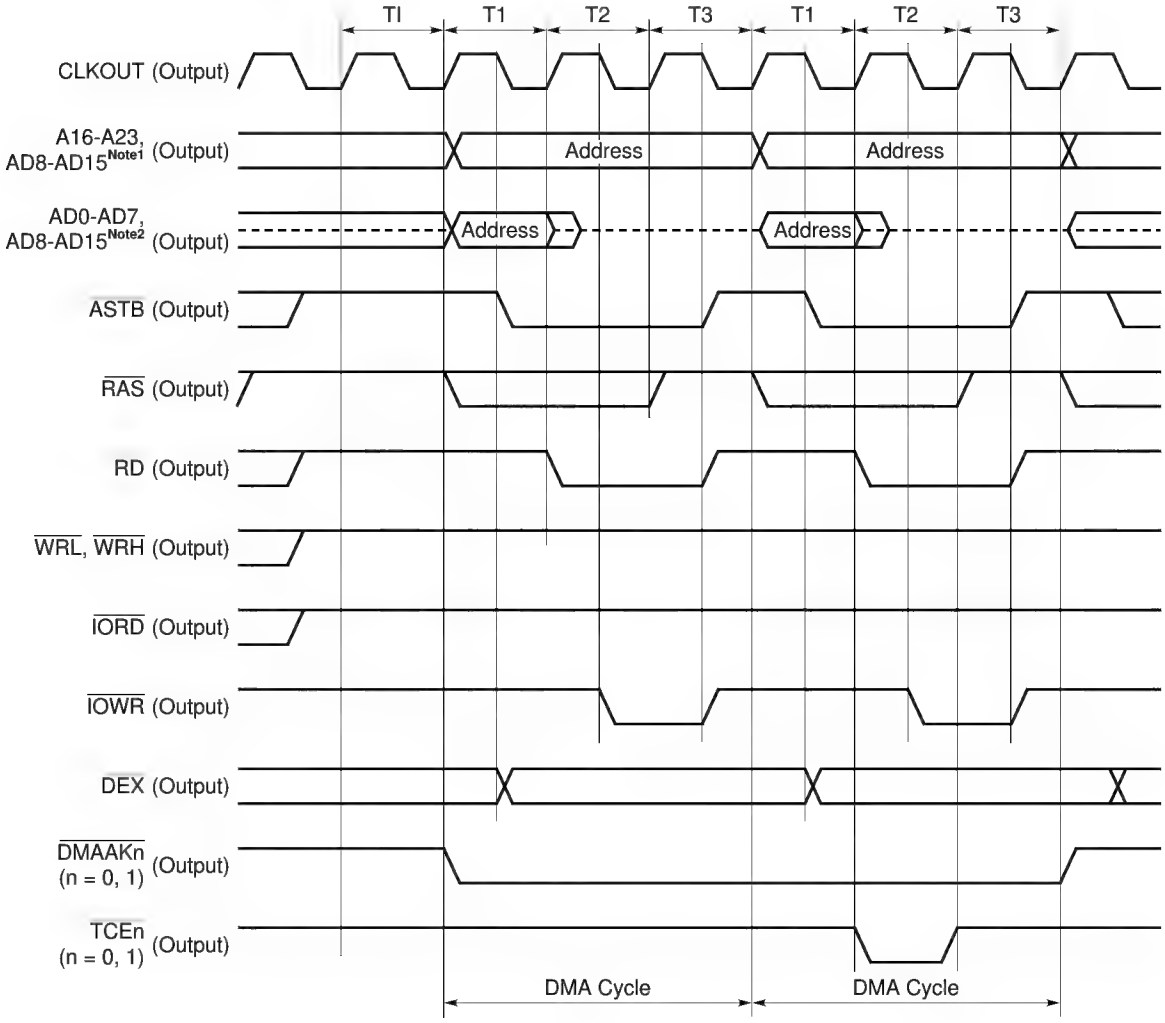
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The broken line indicates high-impedance.

Figure 6-17. Memory → External I/O (2/2)

(b) Demand Release Mode and Burst Mode (Two Transfer)



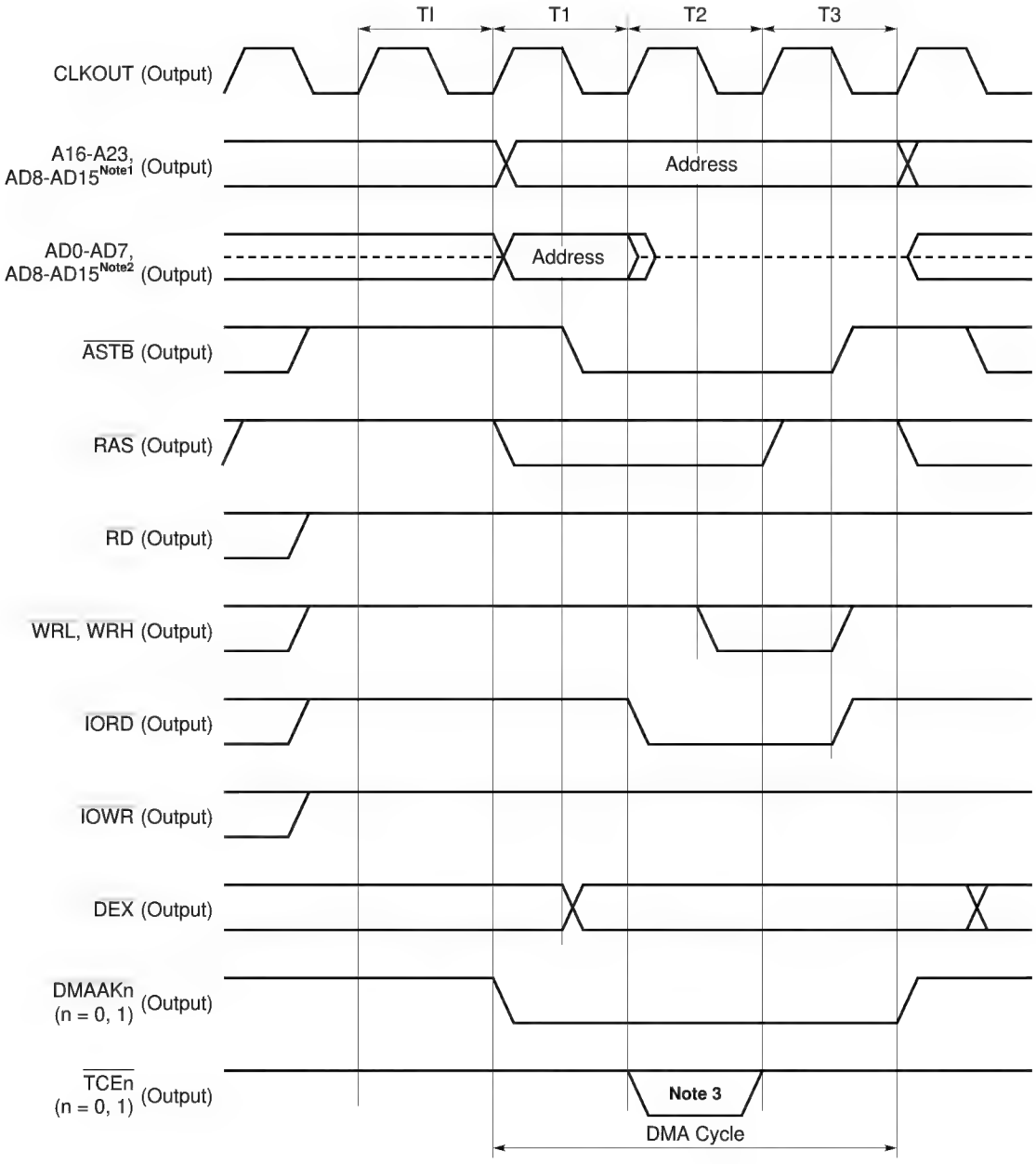
Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

Figure 6-18. External I/O → Memory (1/2)

(a) Single Transfer Mode and Single Step Mode



Notes 1. When the external data bus width is 8-bit:

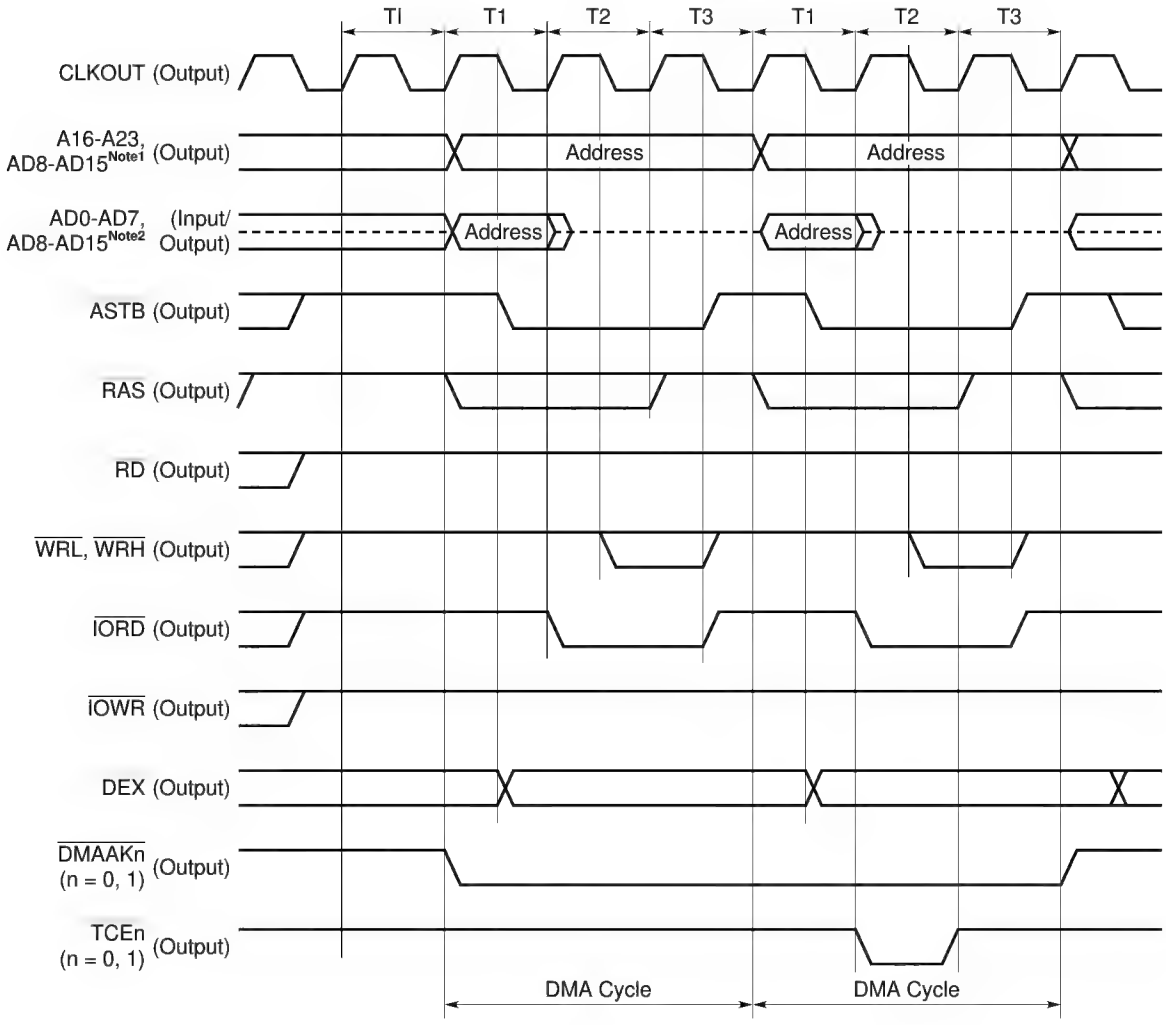
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The broken line indicates high-impedance.

Figure 6-18. External I/O → Memory (2/2)

(b) Demand Release Mode and Burst Mode (Two Transfer)

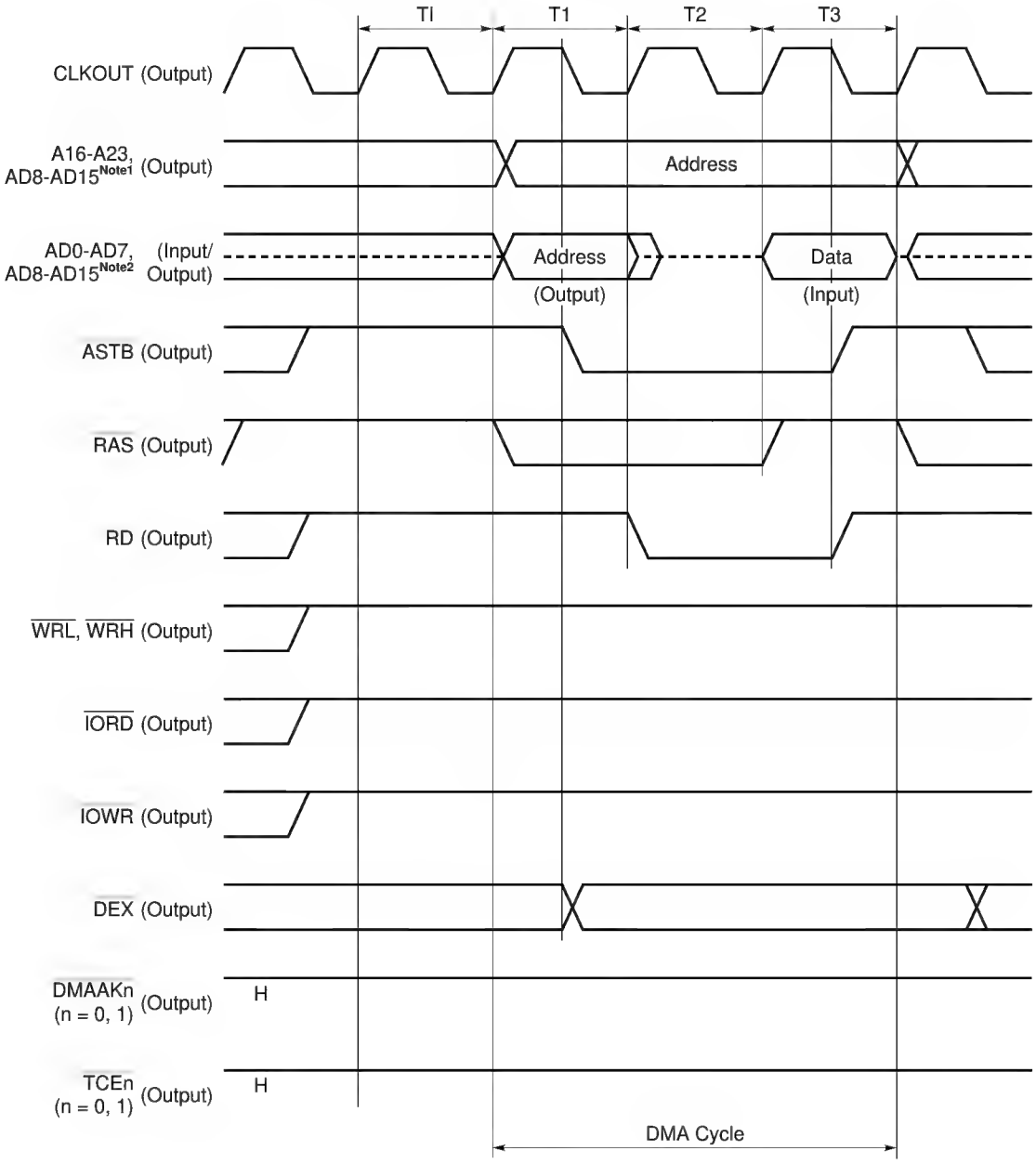


Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

Figure 6-19. Memory → Internal I/O (SFR)

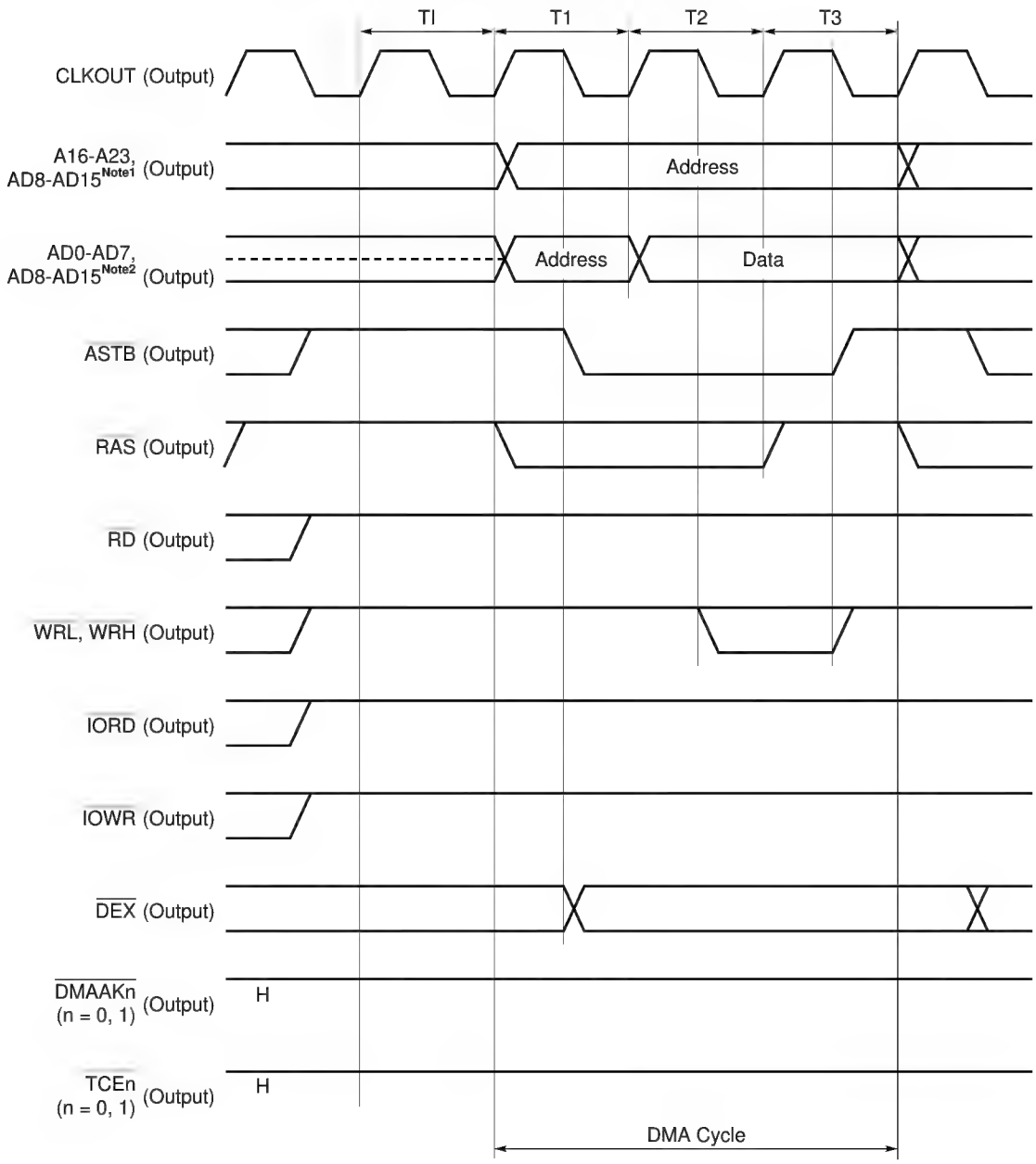


Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

Figure 6-20. Internal I/O (SFR) → Memory



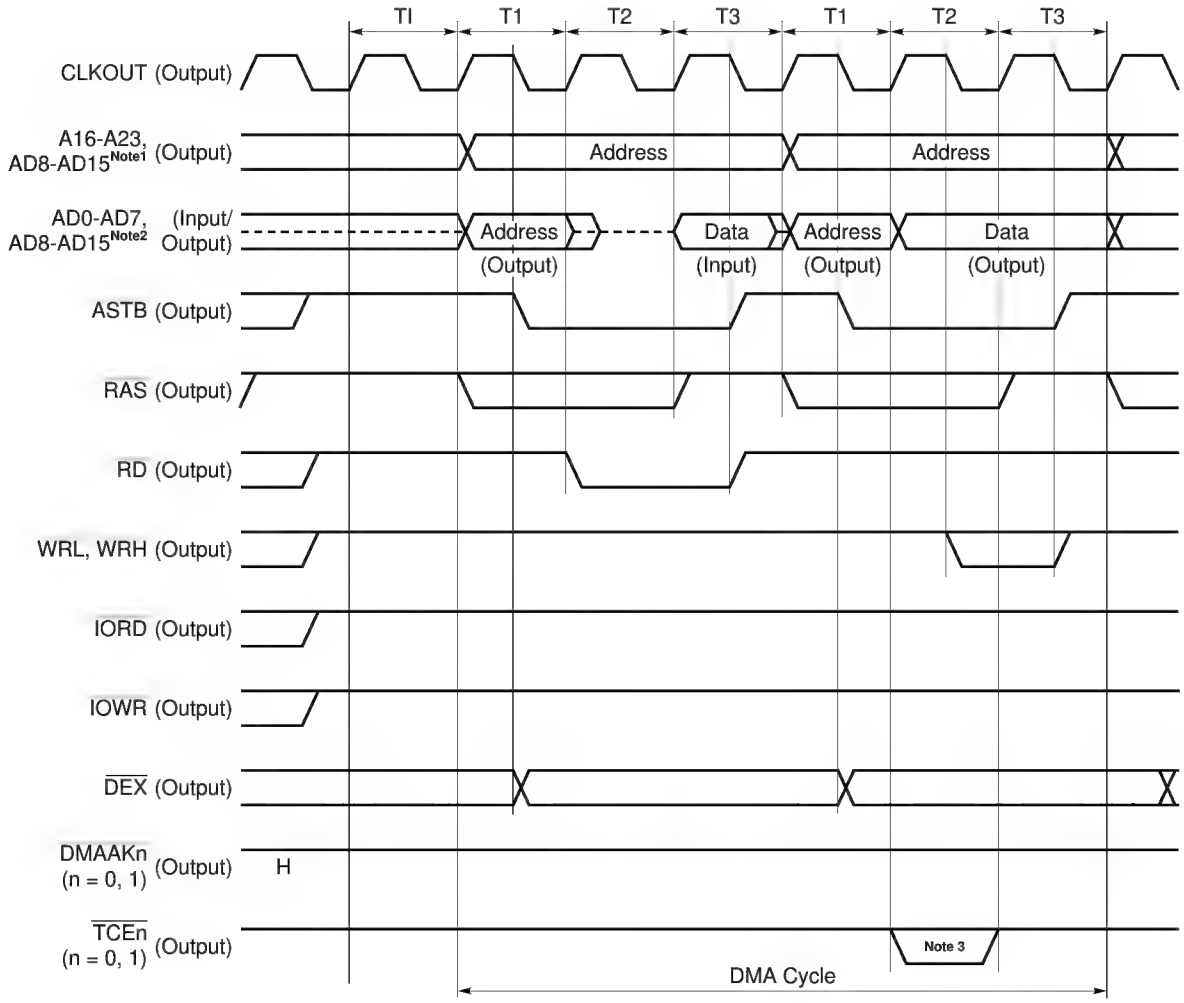
Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

Figure 6-21. Memory ↔ Memory (1/2)

(a) Single Transfer Mode and Single Step Mode



Notes 1. When the external data bus width is 8-bit:

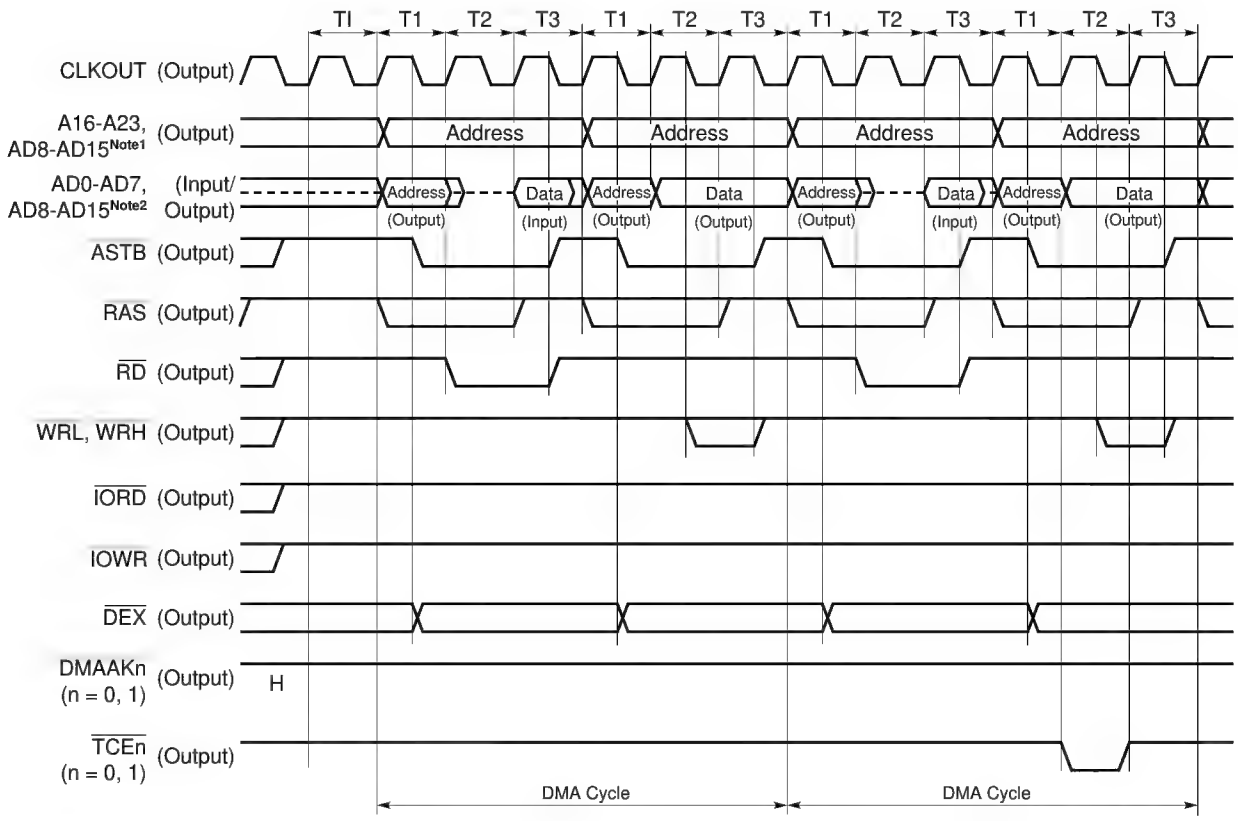
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The bloken line indicates high-impedance.

Figure 6-21. Memory ↔ Memory (2/2)

(b) Burst Mode (Two Transfer)



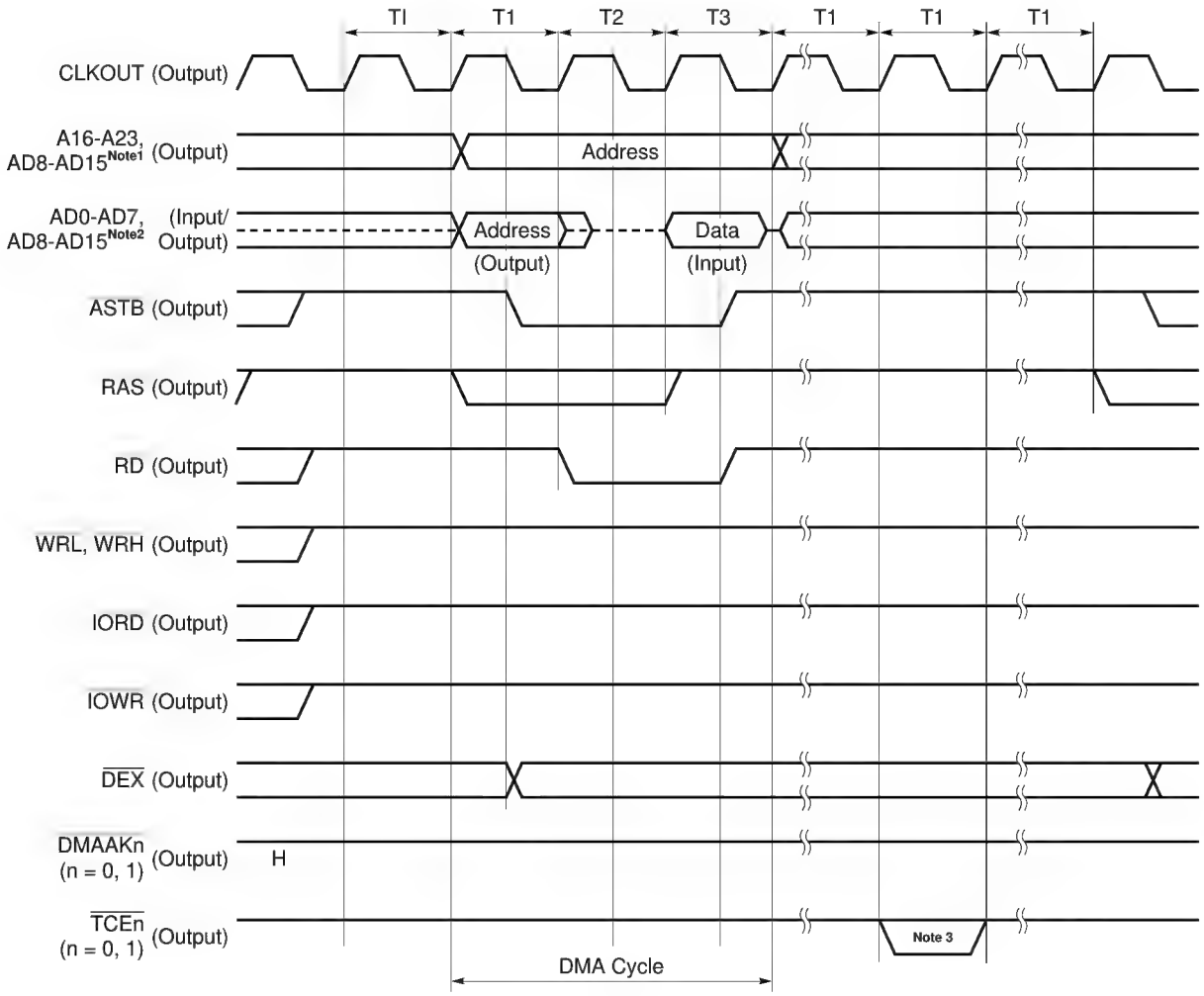
Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

Figure 6-22. Memory → SFR (Memory Access) (1/2)

(a) Single Transfer Mode and Single Step Mode



Notes 1. When the external data bus width is 8-bit:

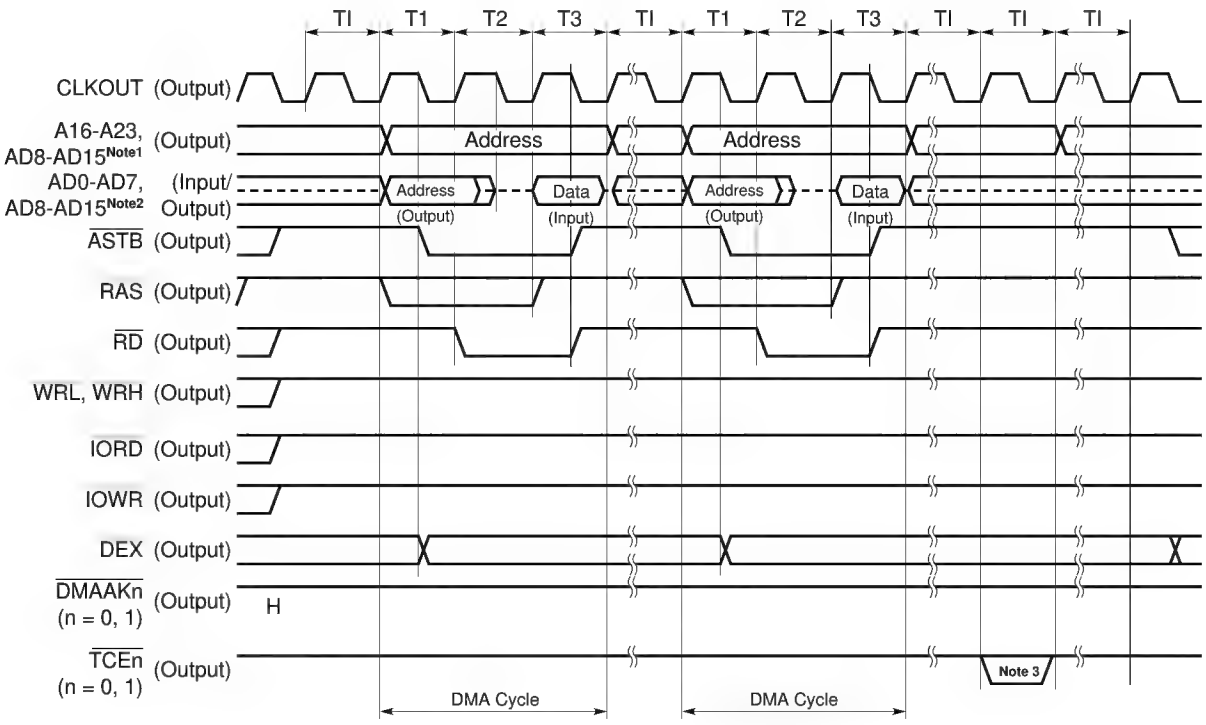
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The broken line indicates high-impedance.

Figure 6-22. Memory → SFR (Memory Access) (2/2)

(b) Burst Mode (Two Transfer)



Notes 1. When the external data bus width is 8-bit:

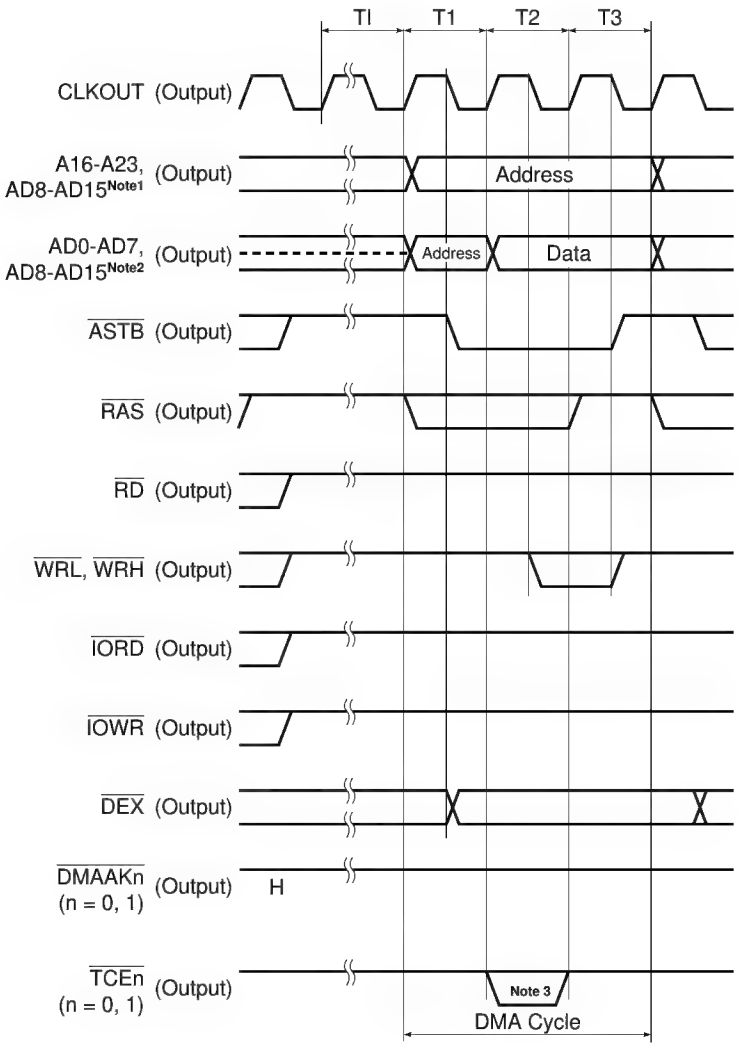
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The broken line indicates high-impedance.

Figure 6-23. SFR (Memory Access) → Memory (1/2)

(a) Single Transfer Mode and Single Step Mode



Notes 1. When the external data bus width is 8-bit:

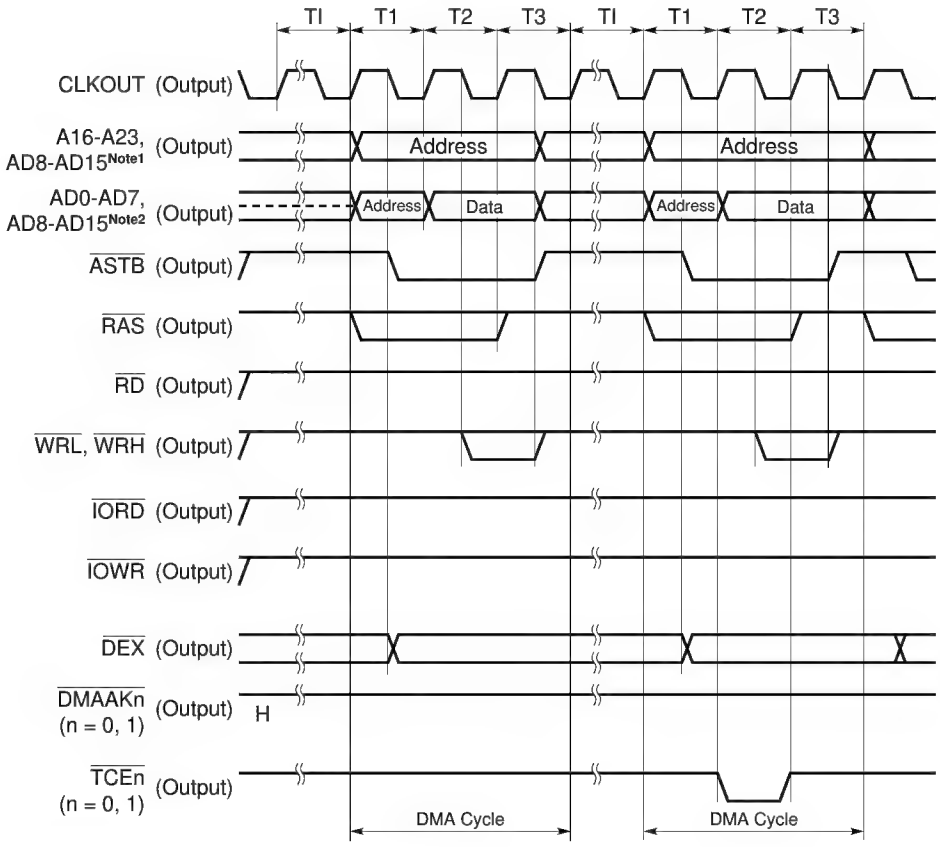
2. When the external data bus width is 16-bit:

3. The low-level signal is output when a TC borrow is generated.

Remark The broken line indicates high-impedance.

Figure 6-23. SFR (Memory Access) → Memory (2/2)

(b) Burst Mode (Two Transfer)



Notes 1. When the external data bus width is 8-bit:

2. When the external data bus width is 16-bit:

Remark The broken line indicates high-impedance.

6.11 Caution

6.11.1 Restrictions on transfer object specification with main channel DMA controller

In the following DMA operation modes in which the main channel is used, the restrictions described below must be noted when an SFR or memory is specified as the object of a transfer.

- Intelligent DMA mode-1
- Intelligent DMA mode-2
- Next address specification mode
- Memory-to-memory transfer mode

(1) SFRs which can be specified as transfer object

Only certain SFRs can be specified as the object of a DMA transfer (see **6.3.2 Transfer target** (3) SFR). If another SFR (interrupt request control register, interrupt mask flag register, in-service priority register, DMA-related control register, etc.) is specified as the object of a DMA transfer, subsequent normal operation cannot be guaranteed.

(2) Relationship to specification by macro service

It is not possible to specify the same SFR or memory simultaneously as the transfer object of a macro service and a DMA transfer.

(3) Specification when external 8-bit bus mode is used

When the external 8-bit bus mode is used, it is not possible to execute a DMA transfer between a 16-bit SFR (see **Table 3-5 List of Special-Function Registers**) and external memory.

The relationship between the external data bus width and DMA transfer objects is shown in Table 6-7.

Table 6-7. External Data Bus Width and DMA Transfer Objects

Transfer Object	Byte/ Word	Even Address/ Odd Address	External Bus Data Width	
			8 Bits	16 Bits
Memory ↔ I/O	Byte		Usable	Unable
	Word	Even address	Not usable	Usable
		Odd address	Not usable	Not usable
Memory ↔ SFR	Byte		Usable	Usable
	Word	Even address	Not usable	Usable
		Odd address	Not usable	Not usable
Memory ↔ Memory	Byte		Usable	Usable
	Word	Even ↔ Even	Not usable (2-byte transfers used)	Usable
		Even ↔ Odd	Not usable (2-byte transfer used)	Not usable (External memory: used with even address)
		Odd ↔ Even	Not usable (2-byte transfers used)	Not usable (External memory: used with even address)
		Odd ↔ Odd	Not usable (2-byte transfers used)	Not usable (External memory: used with even address)

6.11.2 Restrictions on use of 2-channel operating mode

(1) DMA control register (DMAC0/DMAC1) specification

The DMAC0 and DMAC1 registers are common to the main channel and sub-channel. The following restrictions therefore apply:

TMD1 and TMD0 bits (bits 7 and 6) : The transfer mode specification bits.
The same transfer mode applied to the main channel and sub-channel.

W bit (bit 4) : The transfer processing method specification bit.
The same processing method applies to the main channel and sub-channel.

DIR bit (bit 5) : The transfer direction specification bit.
This bit specifies the main channel transfer direction. The sub-channel transfer direction is determined by the TS1 and TS0 bits (bits 0 and 1) of the DMAM register which specify the main channel and sub-channel directions as being the same or opposite.

(2) DMA control pins

DMA control pins are used with the main side of each channel taking priority. If the main-side DMA request source of one channel does not use a pin, it can be used as the control pin of the sub side of the other channel.

Examples 1. Channel 0: TS1, TS0 bits specified as 01
→ Main channel: DMA request from external DMARQ0 pin
Sub-channel: DMA request from external DMARQ1 pin
Channel 1: TS1, TS0 bits specified as 10
→ Main channel: DMA request from external DMARQ1 pin
Sub-channel: DMA request from external DMARQ0 pin

In this case, because the main channels of channel 0 and channel 1 use the DMA control pin, the sub-channels of channel 0 and channel 1 cannot use the DMA control pin.

Examples 2. Channel 0: TS1, TS0 bits specified as 01
→ Main channel: DMA request from external DMARQ0 pin
Sub-channel: DMA request from external DMARQ1 pin
Channel 1: TS1, TS0 bits specified as 00
→ Main channel: DMA request by serial channel 1 reception interrupt (INTSR1)
Sub-channel: DMA request by serial channel 0 transmission interrupt (INTST0)

In this case, because the channel 1 main channel does not use the DMA control pin, the channel 0 sub-channel can use the DMA control pin in the same way as the main channel.

6.11.3 Transfer mode in refresh cycle automatic insertion enable

When an I/O-to-memory DMA transfer is executed in a condition where the refresh cycle automatic insertion is enable (RFM register RFEN bit = 1), use the single transfer mode. If the demand release mode is specified, the final DMA transfer cycle following the falling of DMARQ input may not be generated correctly.

6.11.4 UART receive interrupt request used as a DMA transfer request source

When a UART reception is executed using the DMA function, even if an error occurs, the DMA transfer is executed from the receive buffer (RxB) to an external memory according to the receive interrupt request. In this case, take precautions against the receive error interrupt which is generated after the DMA transfer.

Data in which a receive error has occurred is stored in the address represented as MAR + TC + 1 (decrement) or MAR – TC – 1 (increment) in the external memory space.

When a receive data in which an error has occurred is manipulated in the receive error interrupt, set the DMA to the disable state to prevent the next DMA transfer from being generated.

[MEMO]

CHAPTER 7 SERIAL INTERFACE FUNCTIONS

The V55PPI is equipped with a 2-channel serial interface unit (ch0, ch1).
The two communication protocols supported by the V55PI are as follows:

(1) Asynchronous UART
(2) Clocked CSI
- SBI: 2-wire serial bus interface
- IOE: I/O expansion 3-wire serial interface

7.1 Features

- Two communication protocols supported
- Two serial channels : 2 channels
- Wake-up function
- On-chip dedicated baud rate generator
- DMA request generated by completion of transmission/reception (transmit/receive data DMA transfer capability)

7.2 Communication Protocols

The UART is an asynchronous serial interface which achieves data synchronization by means of start/stop bits, and is functionally enhanced compared with previous single-chip microcomputer UART functions.

The CSI (clocked serial interface) is a clocked serial interface which achieves synchronization by transmission/reception of a clock. The CSI is a subset of the standard serial bus interface specification for NEC single-chip microcomputers, and I^2C functions are not supported. Also, the wake-up release function is implemented by using macro service.

Table 7-1. Supported Protocols

Serial Interface Unit	Protocol		
	Clocked (CSI)		Asynchronous (UART)
	SBI	IOE	
Channel 0	√	√	√
Channel 1	–	√	√

The UART function or CSI function can be programmably selected for each channel. Protocol selection is performed by means of the protocol selection register (ASP)

7.2.1 UART

(1) Basic configuration

- Control pins
  - Serial data input pin (RxD)
  - Serial data output pin (TxD)
  - Transmission enable control pin ($\overline{\text{CTS}}$)
  - Transmit clock output pin (TxC) ... Channel 0 only
- Transfer control unit, separate 8-bit shift registers for transmission and reception
- UART transmit buffer (TxB)
- Receive buffer (RxB)
- Independent transmit and receive BRG registers (TxBRG, RxBRG)

(2) Features

- Transfer rate: 95 to 390 kbps (with 12.5 MHz = system clock ϕ)
  123 to 500 kbps (with 16 MHz = system clock ϕ)
- Full-duplex operation capability
- On-chip dedicated (transmission and reception) baud rate generators
- Wake-up function
- Zero parity function
- Parity error detection
- Framing error detection
- Overrun error detection
- Three UART interrupt sources
  - UART receive error interrupt (INTSER0, INTSER1)
  - UART reception interrupt (INTSR0, INTSR1)
  - UART transmission interrupt (INTST0, INTST1)
- Macro service function
  - UART reception interrupt (INTSR0, INTSR1)
  - UART transmission interrupt (INTST0, INTST1)

7.2.2 CSI

(1) Basic configuration

- Control pins
  - Serial data input pin (SI/SB0) ... SB0 for channel 0 only
  - Serial data output pin (SO/SB1)... SB1 for channel 0 only
  - Serial clock pin ($\overline{\text{SCK}}$)
- Transfer control unit, common 8-bit shift register for transmission and reception (SIO)
- Receive buffer (RxB)
- Common BRG register for transmission and reception (TxBRG)

(2) Features

- Transfer speed: max. 3.125 Mbps (with 12.5 MHz = system clock ϕ)
  max. 4.0 Mbps (with 16 MHz = system clock ϕ)
- Half-duplex communication
- Data length: 8-bit unit
- External/internal clock selection function
- Data MSB-first/LSB-first selection function
- SBI mode (2-wire NEC type serial bus) ..... ch0 only
  - Address/command/data identification function
  - Function for chip selection by address
  - Wake-up function
  - Acknowledge signal ($\overline{ACK}$) control function
  - Busy signal ($\overline{BUSY}$) control function

7.2.3 Control Register

(1) Protocol selection register (ASP)

This register is used to specify the operating protocol for each channel of the serial interface unit.

The channel pin functions and register functions are changed according to the protocol selected by this register.

This register can be read/written to by an 8/1-bit memory access instruction.

Its value is 00H after a reset.

	7	6	5	4	3	2	1	0
ASP	0	0	0	0	0	0	P (ch1)	P (ch0)

Operation Protocol Specification

P	Operation Protocol
1	Asynchronous (UART)
0	Clocked (CSI)

Remark The shaded area indicates status after a reset

Channel 0 (ch0) mode selection (switching between SBI mode and IOE mode) when using clocked operation is performed by means of the clocked serial interface mode register (CSIM).

(2) Control registers of each protocol

The UART and CSI control registers are mapped onto the same memory addresses. Their functions are changed according to the protocol selected.

Table 7-2. Control Registers

Register address	Selected Protocol	
	Asynchronous P=1	Clocked P=0
0FFF73H	UARTM0 (operating mode)	CSIM0 (operating mode)
0FFF7BH	UARTM1 (operating mode)	CSIM1 (operating mode)
0FFF74H	UARTS0 (status)	SBIC0 (control)
0FFF7CH	UARTS1 (status)	Not used[Note 1]
0FFF75H	TxB0 (transmit buffer)	SIO0 (shift register)[Note 2]
0FFF7DH	TxB1 (transmit buffer)	SIO1 (shift register)[Note 2]
0FFF76H	RxB0 (receive buffer)	RxB0 (receive buffer)
0FFF7EH	RxB1 (receive buffer)	RxB1 (receive buffer)
0FFF72H	PRS0 (prescaler)	PRS0 (prescaler)
0FFF7AH	PRS1 (prescaler)	PRS1 (prescaler)
0FFF70H	TxBRG0 (transmit BRG)	TxBRG0 (transmit BRG)
0FFF78H	TxBRG1 (transmit BRG)	TxBRG1 (transmit BRG)
0FFF71H	RxBRG0 (receive BRG)	Not used[Note 3]
0FFF79H	RxBRG1 (receive BRG)	

Notes 1. The SBIC register is not used in the ch1 3-wire serial I/O mode (IOE mode).

2. In the clocked mode, transmit data is written directly to the transmit shift register (SIO).

3. In the clocked mode, the data transfer rate is the same for transmission and reception. Only the reception side baud rate generator (BRG) is used.

(3) Control pins of each protocol

The UART and CSI control pins are assigned to the same port pins. Their functions are changed according to the protocol selected (when the port is set to the control function (PMC3n = 1).

Table 7-3. Control Pins

Control Pin	Selected Protocol	
	Asynchronous P=1	Clocked P=0
Transmit data	TxD0	SO0/SB0 (ch0)
output pin	TxD1	SO1 (ch1)
Receive data	RxD0	SI0/SB1 (ch0)
output pin	RxD1	SI1 (ch1)
Clock pin	TxC	$\overline{SCK0}$ (ch0)
Transmission control pin	$\overline{CTS0}$	Not used (ch0)
Clock/control pin	$\overline{CTS1}$	$\overline{SCK1}$ (ch1)

7.3 Baud Rate Generator

The V55PI incorporates baud rate generators (BRGs) which generate a dedicated transmission serial clock and a dedicated reception serial clock.

With the UART, therefore, it is possible to set any operating baud rate for transmission and reception independently (with the CSI, there is a single baud rate generator for transmission and reception).

7.3.1 Configuration

The baud rate generators are dedicated 8-bit timers which generate the serial clocks for transmission and reception.

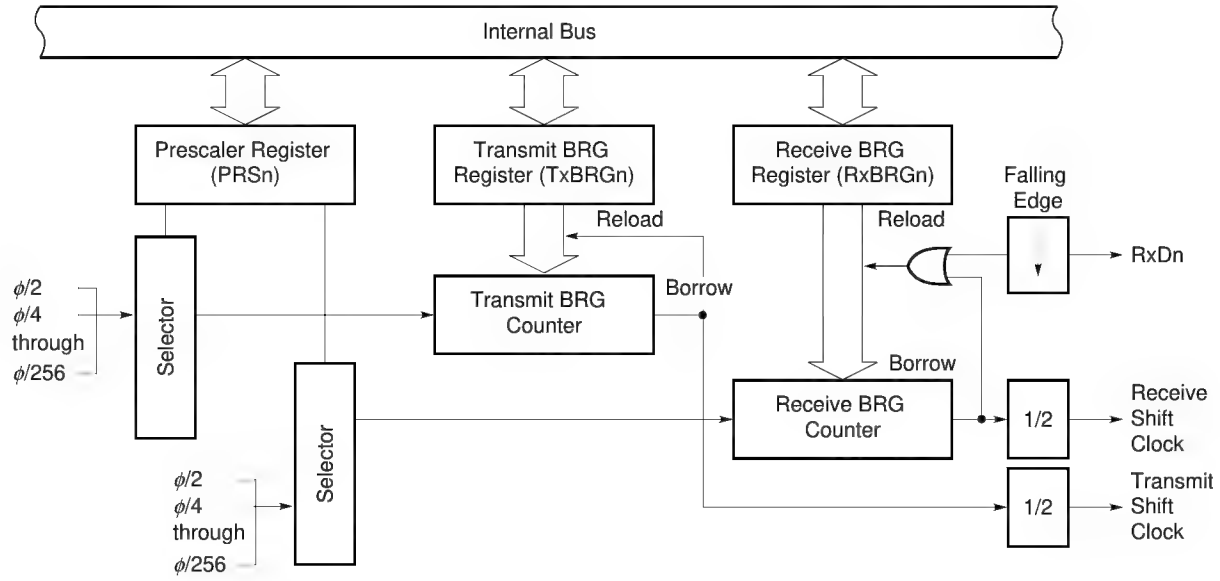
The transmit/receive BRG counter borrow outputs are divided by 2 and used as serial clocks.

One BRG counter cycle is specified by setting the counter value (0 to 255) in the BRG register (TxBRG/RxBRG) in the program.

Scaled output of the internal system clock is specified by the prescaler register (PRS) for the input clock of each BRG counter.

The baud rate generator block diagram is shown in Figure 7-1.

Figure 7-1. Baud Rate Generator Block Diagram



Remark n=0 or 1

7.3.2 Operation

Baud rate generator operation depends on the selected protocol as follows.

(1) Operation with UART

In the receive BRG, a value is preset from the BRG register (RxBRG) into the receive BRG counter after a start bit is detected, and receive data synchronization is established with the serial clock.

The transmit BRG generates a serial clock without regard to the above receive operation.

The system clock ϕ is used for sampling (start bit detection) of the serial receive data RxD in the UART.

Caution With the UART, a value of 7 or above should be set in the receive BRG register to minimize phase discrepancy between the serial clock and the receive data.

(2) Operation with CSI

The receive BRG is not used.

If an internal clock is selected as the serial clock, the transmit BRG output clock is used as the serial clock for transmission and reception.

The expression for calculating the baud rate is shown below.

[Baud rate calculation expression]

$$\frac{\phi}{2^n} \times \frac{1}{(M+1)} \times \frac{1}{2} = \text{serial clock frequency}$$

$$\text{Baud rate} = \frac{\phi}{2^{n+1}\,(M+1)} \text{ [bps]}$$

ϕ : System clock frequency [Hz]

M : 8-bit BRG register set value
($0 \leq M \leq 255$; with UART: $7 \leq M \leq 255$)

$\phi/2^n$: Input clock to BRG selected by prescaler register (PRS)
($1 \leq n \leq 8$; n: set value for the corresponding RPS register bit + 1)

The set value in the BRG register (TxBRG/RxBRG) corresponding to each standard baud rates with a 12.5 MHz or 16 MHz internal clock and the error (%) are shown in Table 7-4.

Table 7-4. Values Set in BRG Register

(1) ϕ = 12.5 MHz

Baud Rate (bps)	n	BRG register Set Value (M)	Error (%)
150	8	162	–0.15
300	7	162	–0.15
600	6	162	–0.15
1200	5	162	–0.15
2400	4	162	–0.15
4800	3	162	–0.15
9600	2	162	–0.15
19200	1	162	–0.15
38400	1	80	+0.47
76800	1	40	–0.76
153600	1	19	+1.72
390625	1	7	0
1.5625 M	1	1 Note	0
3.125 M	1	0 Note	0

Note Only a value of 7 or above can be set for M with the UART. Therefore M cannot be set to 0 or 1.

(2) ϕ = 16 MHz

Baud Rate (bps)	n	BRG register Set Value (M)	Error (%)
150	8	207	–0.16
300	7	207	–0.16
600	6	207	–0.16
1200	5	207	–0.16
2400	4	207	–0.16
4800	3	207	–0.16
9600	2	207	–0.16
19200	1	207	–0.16
38400	1	103	–0.16
76800	1	51	–0.16
153600	1	25	–0.16
500000	1	7	0
1.0 M	1	3 Note	0
4.0 M	1	0 Note	0

Note Only a value of 7 or above can be set for M with the UART. Therefore M cannot be set to 0 or 3.

7.3.3 Control registers

(1) Prescaler registers (PRS0, PRS1)

These are 8-bit registers which specify the baud rate generator input clock.

These registers can be read/written to by an 8/1-bit memory manipulation instruction (PRS0 for ch0, PRS1 for ch1).

Their value after a reset is 00H.

	7	6	5	4	3	2	1	0
PRS0	0	0	TSK2	TSK1	TSK0	RSK2	RSK1	RSK0
PRS1								

Input clock specification to the receive serial clock generator

RSK2	RSK1	RSK0	Input Clock Frequency
0	0	0	$\phi/2$
0	0	1	$\phi/2^2$
0	1	0	$\phi/2^3$
0	1	1	$\phi/2^4$
1	0	0	$\phi/2^5$
1	0	1	$\phi/2^6$
1	1	0	$\phi/2^7$
1	1	1	$\phi/2^8$

Input clock specification to the transmit serial clock generator

TSK2	TSK1	TSK0	Input Clock Frequency
0	0	0	$\phi/2$
0	0	1	$\phi/2^2$
0	1	0	$\phi/2^3$
0	1	1	$\phi/2^4$
1	0	0	$\phi/2^5$
1	0	1	$\phi/2^6$
1	1	0	$\phi/2^7$
1	1	1	$\phi/2^8$

ϕ : Internal system clock frequency

Remark The shaded area indicates status after a reset.

(2) Transmit baud rate generator registers (TxBRG0, TxBRG1)

These are 8-bit registers used to set the transmit baud rate count value. These registers can be written to or read by an 8/1-bit memory manipulation instruction.

TxBRG0 is the register for channel 0, and TxBRG1 the register for channel 1. In the clocked serial interface mode, these registers perform baud rate specification for both transmission and reception.

These registers are undefined after a reset.

(3) Receive baud rate generator registers (RxBRG0, RxBRG1)

These are 8-bit registers used to set the receive baud rate count value. These registers can be written to or read by an 8/1-bit memory manipulation instruction.

RxBRG0 is the register for channel 0, and RxBRG1 the register for channel 1. These registers are not used in the clocked serial interface mode.

These registers are undefined after a reset.

Caution The transmit/receive BRG register should not be read during transmission/reception. Otherwise the shift clock cycle will be temporarily disrupted.

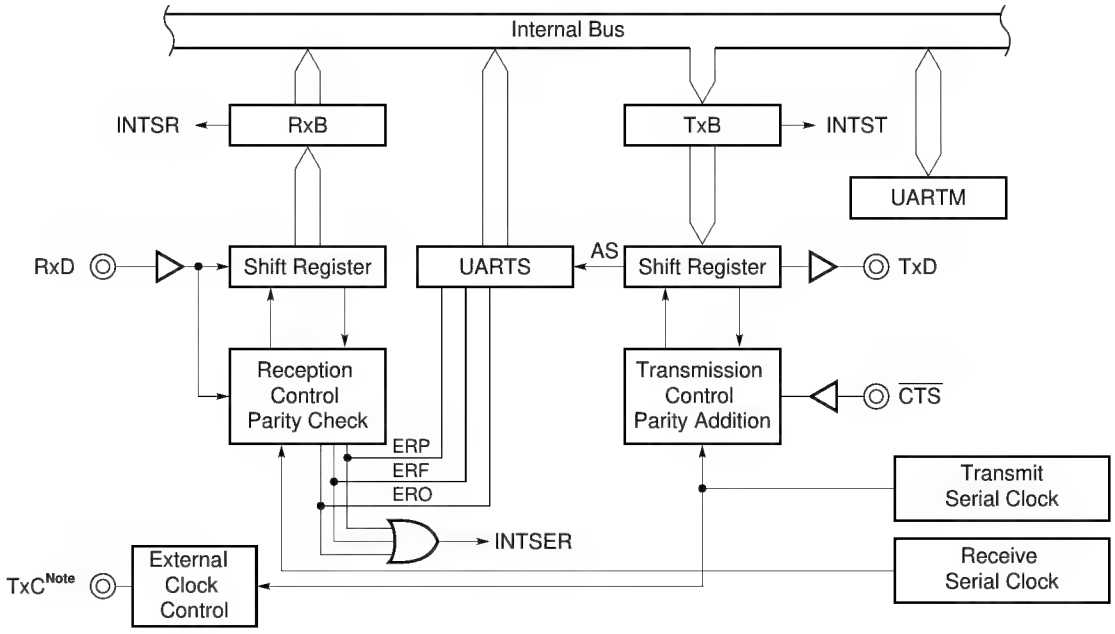
7.4 UART Operation

7.4.1 Configuration

The V55PI is provided with a UART which incorporates a dedicated baud rate generator. The UART block diagram is shown in Figure 7-2.

The UART comprises a serial data input pin (RxD), serial data output pin (TxD), serial clock output pin[Note] (TxC), transmission enabled status control input pin ($\overline{CTS}$), a transfer control unit, 8-bit shift registers for transmission and reception, a UART mode register (UARTM), UART status register (UARTS), UART transmit buffer (TxB), receive buffer (RxB), and a baud rate generator for transmission and reception. Since separate shift registers and buffers are provided for transmission and reception, transmission and reception can be performed independently (full-duplex operation).

Figure 7-2. UART Block Diagram



Note Channel 0 only

7.4.2 Transmit operation

(1) Enabled state

Setting: UART mode register (UARTM) TxE bit = 1 and $\overline{CTS}$ pin = 0

Operation: <1> Data is transmitted from the shift register via the TxD pin.

<2> When all the data of one frame is transmitted, the data of the TxB buffer is transferred to the shift register and transmit interrupt is generated.

<3> Next, the contents of the shift register is transferred to the TxD pin.

(2) Disabled state

Setting: $\overline{CTS}$ pin = 1

or the TxE bit of the UARTM register = 0

Operation: Data transfer from the TxB buffer to the shift register is disabled (Same when the shift register is empty.)

Caution When a transmission disabled operation is performed during transmission, all the data to be transmitted will be transmitted to the end.

(3) Start-up of transmission

There are two methods for starting up transmit. They are as follows.

- Setting the transmission enabled state when the UART transmit buffer is empty.
  The UART transmit interrupt request is generated. In the interrupt process, the transmit data is written in the transmit buffer.

- Writing the transmit data in the transmit buffer in the transmission disabled state and then setting in the transmission enabled state later.
  The data stored in the transmit buffer is transmitted.

If the transmission enabled state is already set, the transmit data is written in the transmit buffer.
The transmit buffer is made empty by the $\overline{RESET}$ input. When the transmission enabled state is set in this state, the UART transmit interrupt request is generated.

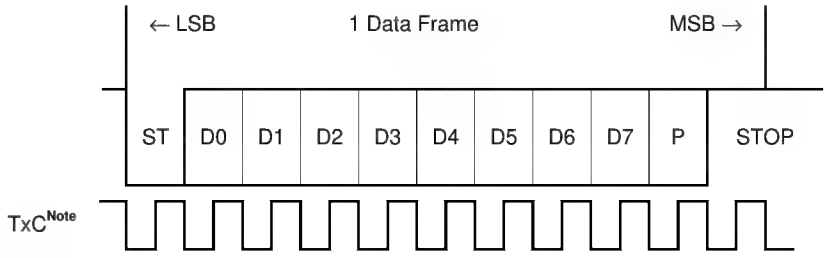
(4) Port control and differences

When the $\overline{CTS}$ pin is set to the port mode (PMC33 = 0 or PMC36 = 0) using the port 3 mode control register (PMC3) transmission enabled/disabled can be controlled by just setting the TxE bit of the UARTM register regardless of the $\overline{CTS}$ pin value.

(5) Basic data format

As shown in Figure 7-3, with the basic transmit data format one data frame comprises a start bit, character bits, parity bit and stop bit(s).

Figure 7-3. Basic Data Format



Start bit	(ST)	:	1 bit
Character bits	(D0 to D7)	:	7 or 8 bits
Parity bit	(P)	:	Odd, even, 0, or no parity
Stop bits	(STOP)	:	1 or 2 bits

Note Channel 0 only

7.4.3 Receive operation

(1) Enabled state

Setting: UART mode register (UARTM) RxE bit = 1

Cautions 1. However, when UARTM register WUP bit = 1, the reception enabled stated is set regardless of the RxE bit value (See 7.4.4 Wake-up function).

2. When the reception of the RxD pin input is enabled in the low level state, the low level is determined as the start bit and reception can be started. Therefore, always enable the reception of the RxD pin input in the high level state.

Operations: <1> When the RxD pin input falling signal edge is detected, the receive BRG counter is reset, down count is started, and reception is started.

<2> When all the data of one frame is received, the data in the shift register is transferred to the RxB buffer and receive interrupt is generated.

(2) Disabled state

Setting : UARTM register RxE bit = 0

Operation : Receive data is not input to the shift register.

Caution When the receive disabled operation is performed while data is received, the data being received will not be received properly.

(3) Parity function

When "0 parity", which is a parity funciton, is selected with the PRTY0 and PRTY1 bit of the UARTM register, the transmit data is added with a "0" unconditionally as the parity bit during the transmission of serial data. During the reception of serial data, data is received regardless of the state of the parity added to the received data and parity error is not generated. The "0 parity function" is useful for communication when the serial data transmit/receive format is not fixed, such as when the power of the system is turned on.

7.4.4 Wake-up function

UART receive interrupt generation cna be controlled with the wake-up bit (WUPT) value transmitted after the serial data character bit.

(1) Setting

By setting the WUP bit of the UART mode register (UARTM) to 1, the wake-up function can be selected.

(2) Operation

If data is transmitted when the wake-up function is selected, the data can be received even if the receiving side is set to the receive disabled state.

Next, the operations in the receive disabled state (RxE bit = 0) and enabled state (RxE bit = 1) is explained.

- **Receive disabled state (RxE bit = 0)**
  - <1> The receive shift operation is synchronized with the detection of the start bit and started.
  - <2> When the data of one data frame is received, the receive data wake-up bit data is checked whether it is "1".
  - <3> If "1", the contents of the receive shift register is transmitted to the receive buffer and receive interrupt or receive error interrupt is generated.

- **Receive enabled state (RxE bit = 1)**
  - <1> The receive shift operations is synchronized with the detection of the start bit and started.
  - <2> When the data of one data frame is received, the contents of the receive shift register is transferred to the receive buffer and receive interrupt or receive error interrupt is generated.

(3) Features

By using the wake-up function, several systems can be connected together by UART as shown in Figure 7-4 to enable data to be transmitted and received more efficiently.

When the slave device is set to the wake-up state when receiving the WUPT bit = 0 data, serial receive interrupt is generated for the released slave device only. Therefore, when performing transmission and reception with the specified slave device, other slave devices can operate regardless of serial communication.

If the wake-up function is not equipped, all the slave devices are affected in each serial communication because interrupt is generated each time data is received.

(4) Example of use

When transmitting data to a device with a certain address, with the V55PI as the master.

<1> The WUP bit of the UARTM register is set to 1 and the RxE bit is set to 0 for both the master side and several slave sides.

<2> Master side: The WUPT bit is set to 1 and the address data is transmitted.

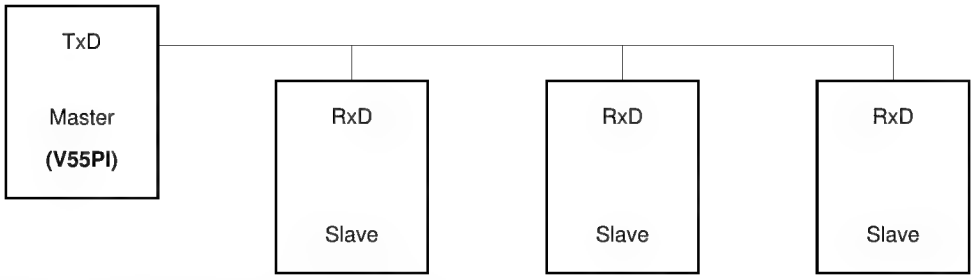
Slave side (several): As the wake-up function is selected, reception interrupt is generated and the transmitted address data is checked by the software.

<3> Specified slave: When the slave detects that it has the specified address, the RxE bit of the UARTM register is set to 1.

Master side: The WUPT bit of the data is set to 0 and transmitted.

<4> As the WUPT bit of the transmit data is 0, interrupt is generated only in the slave with the specified address.

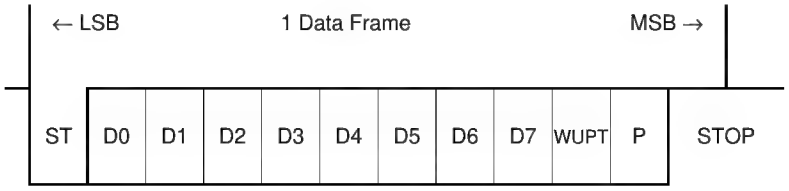
Figure 7-4. Example of Connection of Multiple Systems



(5) Data format when wake-up function is selected

As shown in Figure 7-5, when the wake-up function is selected one data frame comprises a start bit, character bits, wake-up bit (added transmit bit data), parity bit and stop bit(s).

Figure 7-5. Data Format When Wake-Up Function is Selected



Start bit	(ST)	:	1 bit
Character bits	(D0 to D7)	:	8 bits
Wake-up bit	(WUPT)	:	1 bits
Parity bit	(P)	:	Odd, even, 0, or no parity
Stop bits	(STOP)	:	1 or 2 bits

Caution The wake-up bit data is included in the parity calculation data.

7.4.5 Control registers

UART control registers comprise the UART mode register (UARTM0, UARTM1), UART status register (UARTS0, UARTS1), UART transmit buffer (TxB), and UART receive buffer (RxB).

(1) UART mode register (UARTM0, UARTM1)

The individual bits of the UART mode register specify the operating mode as shown below.

After a reset, the value of this register is 00H.

UARTM0
UARTM1

7	6	5	4	3	2	1	0
TxE	RxE	PRTY1	PRTY0	CL/WUPT	SL	ASIE	WUP

Wake-up Operation Enable Specification

WUP	Specification
0	Not Used
1	Used

AS Interrupt[Note] Function Operation Enable Specification

ASIE	Specification
0	Interrupts disabled
1	Interrupts enabled

Stop Bit Length Specification

SL	Stop Bit Length
0	1
1	2

Character Length Specification

WUP	CL/WUPT	Character Length	Wake-Up Bit
0	0	7 Bits	Not Used
0	1	8 Bits	Not Used
1	0	8 Bits	0 Sent
1	1	8 Bits	1 Sent

Parity Bit Specification Operation

PRTY1	PRTY0	Operation
0	0	No Parity
0	1	0 Parity
1	0	Odd Parity
1	1	Even Parity

Reception Enable Specification

RxE	Specification
0	Reception Disabled State
1	Reception Enabled State

Transmission Enable Specification

TxE	$\overline{CTS0}$	Specification
0	×	Transmission Disabled State
1	0	Transmission Enabled State
1	1	Transmission Disabled State

Note AS (All Sent) interrupt: All Sent state detection interrupt (Refer to **7.4.6 (3)**)

Caution With the exception of the ASIE flag, UARTM register bit data cannot be changed during UART transmission/reception.

Remark The shaded area indicates status after a reset

CL/WUPT Character Length or Wake-Up Bit Data Specification

The CL/WUPT bit function depends on the value of the WUP bit (wake-up function selection bit). When the parity function is used, the transmission wake-up bit data is included in the transmission parity calculation data.

(a) When wake-up function is not used (WUP = 0)

CL is valid and specifies the character length of one data frame.

CL	Character Length
0	7 bits
1	8 bits

(b) When wake-up function is used (WUP = 1)

WUPT is valid, and specifies the wake-up bit data to be transmitted. In the wake-up bit mode, the character length is fixed at 8 bits.

WUPT	Wake-Up Bit Data
0	0 sent in transmit data wake-up bit data position
1	1 sent in transmit data wake-up bit data position

PRTY1 PRTY0 Parity Bit Specification

In the case of zero parity, the transmitting side transmits with parity bit 0. The receiving side does not generate a reception parity error.

RxE Reception Enable Specification

Specifies the reception enabled/disabled state. The function depends on the value of the WUP bit (wake-up operation enable selection bit)

(a) When wake-up function is not used (WUP = 0)

RxE	Operation
0	Reception disabled state
1	Reception enabled state

(i) When reception disabled state is set (RxE = 0)

Receive data is not input to the shift register.

(ii) When reception enabled state is set (RxE = 1)

Input of receive data to the shift register is started in synchronization with detection of the start bit, and when one data frame has been received the contents of the shift register are transferred to the receive buffer (RxB). A UART reception interrupt/UART receive error interrupt is generated in synchronization with the transfer from the receive shift register to the receive buffer.

(b) When wake-up function is used (WUP = 1)

(i) When reception disabled state is set (RxE = 0)

The receive data shift operation is started in synchronization with detection of the start bit, and when one data frame has been received a check is performed to see if the receive data wake-up bit data is "1". If it is "1", the contents of the receive shift register are transferred to the receive buffer (RxB) and a UART reception interrupt/UART receive error interrupt is generated, as in (a) (ii).

(ii) When reception enabled state is set (RxE = 1)

The operation is the same as in (a) (ii), except that the receive data wake-up bit data is loaded into the WUPR flag in the UART status register (UARTS).

TxE Transmission Enable Control Specification

If the port which also functions as the $\overline{\text{CTS}}$ pin is set to port mode by the port 3 mode control register (PMC3) (bits 3 & 6 = 0), transmission operation enabling/disabling can be controlled by the TxE bit alone.

(2) UART status register (UARTS0, UARTS1)

The UARTS register indicates the receive error status when one character has been received. The error flag contents are updated each time data is received.

After a reset the value of the UARTS register is 20H.

UARTS0
UARTS1

7	6	5	4	3	2	1	0
AS	WUPR	TxBE	RxBF	0	ERP	ERF	ERO

Overrun Error Flag

ERO	Operation
0	When reception of the next data is completed after data is read from the receive buffer
1	When reception of the next data is completed before the previous data has been taken from the receive buffer

Framing Error Flag

ERF	Operation
0	When a stop bit is detected on completion of the next reception
1	When a stop bit is not detected

Parity Error Flag

ERP	Operation
0	When the transmit parity and receive parity match in the next parity check
1	When the transmit parity and receive parity do not match

Receive Buffer Full Flag

RxBF	Operation
0	When no valid data remains in the receive buffer as the result of reading receive data from the receive buffer
1	When shift register reception is complete and receive data has been transferred to the receive buffer

See the next page.

UARTS0
UARTS1

7	6	5	4	3	2	1	0
AS	WUPR	TxBE	RxBF	0	ERP	ERF	ERO

See the previous page.

Transmit Buffer Empty Flag

TxBE	Operation
0	When the transmit buffer is full as the result of a transmit data write to the transmit buffer
1	When the transmit data in the transmit buffer has been transferred to the shift register and the transmit buffer has become empty

Receive Wake-Up Bit Detection Flag

WUPR	Operation
0	Received Wake-up Bit Data = 0
1	Received Wake-up Bit Data = 1

All Transmit Data Sent End Detection Flag

AS	Operation
0	Transmit data write to the transmit buffer
1	When the last data bit has been sent from the transmission pin when transmission is enabled and the transmit buffer is empty

Caution ERP, ERF and ERO can be read or written to by an instruction. They are reset (0) by writing 0. Other status flags are read-only.

Remark The shaded area indicates status after a reset

ERO Overrun Error Flag

When an overrun error is generated, receive data is still transferred from the shift register to the buffer (RxB). Therefore, the last receive data transferred is written in RxB.

Even if the ERO flag is cleared, if receive data remains in RxB an overrun will occur again when the next data frame is received. Ensure, therefore, that an RxB read is also performed when an overrun error is generated.

ERF Framing Error Flag

The ERF flag is updated each time a data frame is received.

ERP Parity Error Flag

The ERP flag is updated each time a data frame is received.

If the "0 parity" function has been selected, a UART receive error interrupt (INTSER) is not generated by a parity error. When "no parity" or "0 parity" is specified, the ERP flag is undefined. However, the parity operation results can be detected wtih the ERP flag.

Setting	ERP	Operation Results	Interrupt When ERP = 1
No parity	0	When the number of 1 in the data is odd	Non-generated
	1	When the number of 1 in the data is even	
0 parity	0	When the number of 1 in the data is even	Non-generated
	1	When the number of 1 in the data is odd	
Odd parity	0	No error (When the number of 1 in the data is odd)	Generated
	1	Error (When the number of 1 in the data is even)	
Even parity	0	No error (When the number of 1 in the data is even)	Generated
	1	Error (When the number of 1 in the data is odd)	

RxBF Receive Buffer Full Flag

This is a status flag which indicates that valid receive data has been stored in the receive buffer.

By polling this flag it is possible to find out whether a receive data read can be performed.

TxBE Transmit Buffer Empty Flag

This is a status flag which indicates that the transmit buffer is empty.

By polling this flag it is possible to find out whether a receive data read can be performed.

WUPR Wake-Up Bit Data Monitor Flag

As the received wake-up bit data corresponds to the flag status, it is possible to monitor the wake-up bit data. This flag is valid when the wake-up function is specified by the UARTM register.

AS All Transmit Data Sent End Detection Flag

This pin can be used to monitor whether data transmission from the TxD pin has been completed. A possible application is detection of transmission to the circuit from the TxD pin of all data written to TxB using a modem, etc., and subsequent circuit disconnection.

(3) Receive buffers (RxB0, RxB1)

The receive buffers are 8-bit buffers which hold receive data.

In the 7-bit character reception state, "0" is stored in the MSB automatically.

In the reception enabled state, receive data is transferred from the shift register after one receive data frame has been processed.

At the same time, a UART reception interrupt request (INTSR0/INTSR1) is generated.

(4) UART transmit buffers (TxB0, TxB1)

The transmit buffers are 8-bit buffers which hold transmit data.

When a data write to the transmit buffer is performed in the transmission enabled state, data is transferred from the buffer to the shift register in synchronization with the transmit clock, and the transmit operation is started.

If transmit data is written to the transmit buffer in the transmission disabled state, serial transmission is not performed and the data remains in the buffer. If the transmission enabled state is switched to subsequently, the data in the buffer is sent to the shift register and transmission is started.

When data is sent from the transmit buffer to the transmit shift register, a UART transmission interrupt request (INTST0/INTST1) is generated.

7.4.6 Interrupt requests

The following three kinds of interrupt request are generated from the UART:

UART receive error interrupts	: INTSER0, INTSER1
UART reception interrupts	: INTSR0, INTSR1
UART transmission interrupts (dual function as AS state detection interrupts)	: INTST0, INTST1

The order of priority for these three kinds of interrupt request is as follows:

INTSERn > INTSRn > INTSTn (n = 0 or 1)

(1) UART receive error interrupt (INTSER0, INTSER1)

If a receive error is generated in the reception enabled state, a UART receive error interrupt is generated. The UART receive error interrupt is generated after the first stop bit of the receive data is detected, in synchronization with the transfer of the data in the receive shift register to the receive buffer (this is the same generation timing as for the INTSRn reception interrupt).

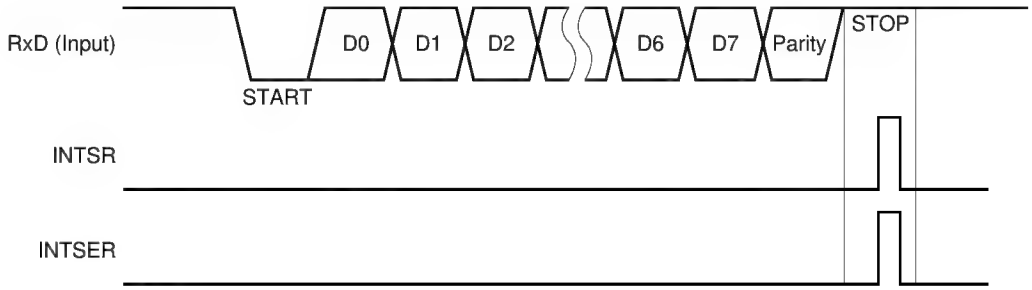
A UART receive error interrupt is not generated in the reception disabled state. Also, if the zero parity function has been selected (PRTY1 = 0, PRTY0 = 1), a UART receive error interrupt is not generated by a parity error.

Table 7-5. Receive Error Causes

Receive Error	Cause
Parity error	The parity specified during transmission and parity of the receive data do not match
Framing error	The stop bit is not detected[Note]
Overrun error	The next data is received before the data from the receive buffer is read

Note In reception, a stop bit operates as only a 1 bit. Therefore, if specified as 2 bits and the 2nd stop bit is low level, error is not generated (acknowledged as the start bit of the next data).

Figure 7-6. UART Reception Error Interrupts Generation Timing



(2) UART reception interrupt (INTSR0, INTSR1)

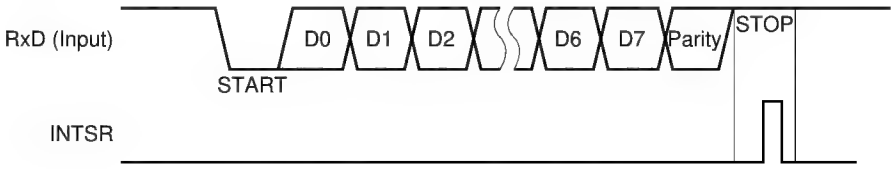
A UART reception interrupt is generated after the first stop bit of the receive data is detected, in synchronization with the transfer of the data in the receive shift register to the receive buffer.

A UART reception interrupt is also generated when a receive error has occurred, but the receive error interrupt is of higher priority in terms of actual servicing. Receive interrupt requests are acknowledged after the receive error interrupt is processed.

If the wake-up condition is not fulfilled when reception is disabled and the wake-up function has been selected (WUP = 1, RxE = 0), a UART reception interrupt is not generated.

If a UART receive interrupt is not generated, the contents of the receive buffer (RxB) are retained.

Figure 7-7. UART Reception Interrupts Generation Timing



Cautions 1. When WUP=1, the reception state is set and a UART reception interrupt/receive error interrupt generated only if RxE = 1 or the receive wake-up bit data = 1.

2. A UART reception interrupt is generated only the first time the condition is fulfilled.

3. If the reception of receive data from the receive buffer (RxB) is late and an overrun error is generated, the receive buffer contents are overwritten with the data received later. In this case, another reception interrupt is generated.

4. The UART reception interrupts share their interrupt response control facilities such as interrupt control register, vector, etc., with the CSI operation serial transmission interrupts (INTCSI0, INTCSI1). Which interrupt control is used is determined by protocol selection.

(3) UART transmission interrupts (INTST0, INTST1)

These interrupts are normally generated when a transmit operation is performed and when the AS state (All Sent state)[Note] detection mode is enabled.

Note The data in the shift register is transferred before the next data is set in the transmit buffer and the no transmit data state (both the transmit buffer and shift register are empty) is set.

(i) In a transmit operation

When the transmit enabled state (TxE = 1) is set and transmission of the last data (STOP bit) of one data frame is completed, and the shift register is detected to be empty, data transfer is performed from the transmit buffer (TxB) to the shift register. When the buffer becomes full as a result, a UART transmission interrupt request is generated.

(ii) When the All Sent state detection mode is enabled (UARTM register: ASIE = 1)

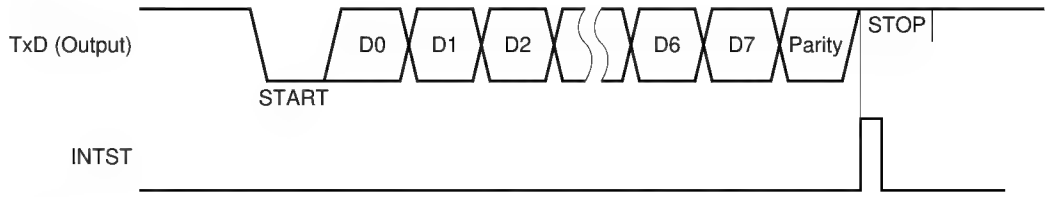
If the transmission enabled state is set and the transmit buffer is empty in addition to the above conditions, a UART transmission interrupt is also generated after transmission of the last data (STOP bit) from the transmission pin TxD.

In this case the AS flag in the UART status register (UARTS) is set (1).

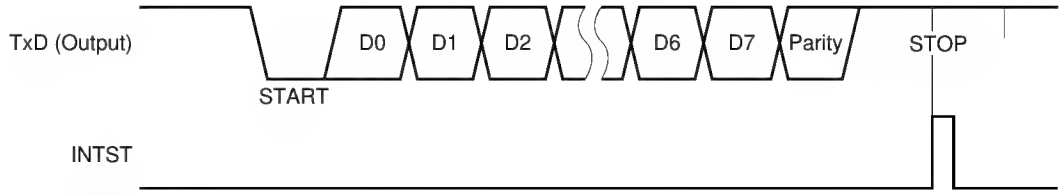
Caution If the UART transmit buffer (TxB) is empty when transmission is enabled, a UART transmission interrupt will be generated. It is therefore possible to generate a transmission interrupt by changing the TxE flag from 0 to 1 after reset release, and to use this to perform data transfer to the transmit buffer by means of interrupt servicing.

Figure 7-8. UART Transmission Interrupt Generation Timing

(a) Stop bit length = 1



(b) Stop bit length = 2



7.4.7 DMA request

The transmit interrupt request and receive interrupt request can be connected to the DMA request as they are. By performing an external memory-to-receive buffer (RxB) transfer and tansmit buffer (TxB)-to-external memory transfer by DMA, CPU loads related to the setting of the transmit data and reception of the receive data can be reduced. As the data is transmitted at high speed, the DMA transmission can be used for high speed serial communication.

Table 7-6 shows the interrupt requests, transfer object, and target DMA channel.

See **6.6.1 DMA mode register (DMAM0, DMAM1)** for how to select DMA.

Cautions 1. When setting the transmit/receive interrupt request as the DMA request, always mask the corresponding interrupt (MK bit of interrupt request control register is 1). If not masked, operations may not be performed properly.

2. In the CST mode, DMA transfer of the transmit/receive data cannot be performed.

Table 7-6. UART Interrupt Requests Connectable to DMA Requests

Serial Interface Channel		Interrupt Requests (DMA Request Source)	Transfer Object I/O	TS1	TS0	Target DMA Channel
ch0	Receive	INTSR0	RxB0	1	1	ch1: Subchannel
	Transmit	INTST0	TxB0	0	0	ch0: Subchannel ch1: Subchannel
				1	1	ch0: Main channel
ch1	Receive	INTSR1	RxB1	0	0	ch1: Main channel
	Transmit	INTST1	TxB1	1	1	ch0: Subchannel

7.4.8 Caution

(1) Transmit disabled operation during transmission

If transmit disabled operation is performed during transmission by setting $\overline{CTS}$ to 1 or TxE to 0, the data being transmitted is transmitted to the end and the transmission is stopped when the data of the transmit shift register has been transmitted.

If the transmit buffer (TxB) contains data, it is not transmitted and is stored until the next transmission enable.

(2) Setting transmit interrupt

When transmission is enabled before the transmit data is written in the transmit buffer, the transmit interrupt request is generated immediately. Always set the interrupt before enabling transmission.

(3) Receive interrupt accompanying receive error interrupt

When reception error interrupt is generated, the receive interrupt request is also generated following it. If the receive interrupt request after the receive error interrupt is not to be acknowledged, clear the RxE bit of the UARTM register. The receive data in the receive buffer (RxB) is transmitted even when errors are generated. When clearing the RxE bit with the software, always read the receive buffer value to prevent overrun errors.

(4) Setting values in receive BRG register

To control phase differences between the serial clock and receive data, set a value above 7 to the receive BRG register for UART.

(5) Reading the transmit/receive BRG register

Do not read the transmit/receive BRG register during transmission and reception as the shift clock period will be upset temporarily.

(6) Transmit/receive interrupt

In the V55PI, two transmit/receive request generations are memorized in the interrupt controller. However, when the interrupt flag is cleared to erase the interrupt request with the software, the two interrupt requests will also be erased.

(7) If transfer is interrupted while in progress

If data transfer is interrupted while in progress, first check the following points.

<1> Is data set in the transfer buffer during transfer interrupt?
A transfer interrupt is generated when TxB becomes empty as a result of transfer data being transferred from the transmission buffer (TxB) to the shift register. Therefore, if setting transfer data with a transfer interrupt, be sure to write data to TxB during the transfer interrupt.

<2> Has the interrupt flag (IF) been cleared by software?
When setting transfer enable, an interrupt request is immediately generated. Therefore, if you set the interrupt request control register (IC××) after setting transfer enable, the interrupt flag may be cleared by error.

See **5.14.4 If an interrupt is not generated**.

7.5 CSI Operation

The V55PI CSI (clocked serial interface) has the following two operating modes.

(1) 3-wire serial I/O mode (IOE mode)

In this mode, 8-bit data transfer is performed using three lines: the serial clock ($\overline{SCK}$), and serial data input and output (SI, SO).

The functions of the V25 and V25+™ have been enhanced, and data MSB-first/LSB-first selection is possible.

(2) Serial bus interface mode (SBI mode)

In the SBI mode, communication is performed with multiple devices by means of two lines: the serial clock ($\overline{SCK}$) and the serial data bus (SB0 or SB1).

This mode conforms to the NEC serial bus format.

In the SBI mode, the sender can output to the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. Thus there is no need for the line for handshaking, allowing input/output ports to be used efficiently.

In addition, wake-up release is performed using macro service.

Caution In the SBI mode, the following settings must be made before enabling reception (CRxE bit = 1) or enabling transmission (CTxE bit = 1):

- **Serial clock selection**
- **SBI mode serial data input/output pin selection**

7.5.1 Configuration

The V55PI's clocked serial interface is configured as shown in Figure 7-9.

The clocked serial interface is composed of the SO0/SB0 (SO1), SIO/SB1 (SI1), and serial clock input/output ($\overline{SCK0}$, $\overline{SCK1}$) pins, clocked interface mode registers (CSIM0, CSIM1), serial interface control register (SBIC0), clocked serial I/O shift registers (SIO0, SIO1), and receive buffers (RxB0, RxB1).

The SO0/SB0 pin functions as the serial data output pin (S0) in the IOE mode, and as the serial data input/output pin SB0 in the SBI mode.

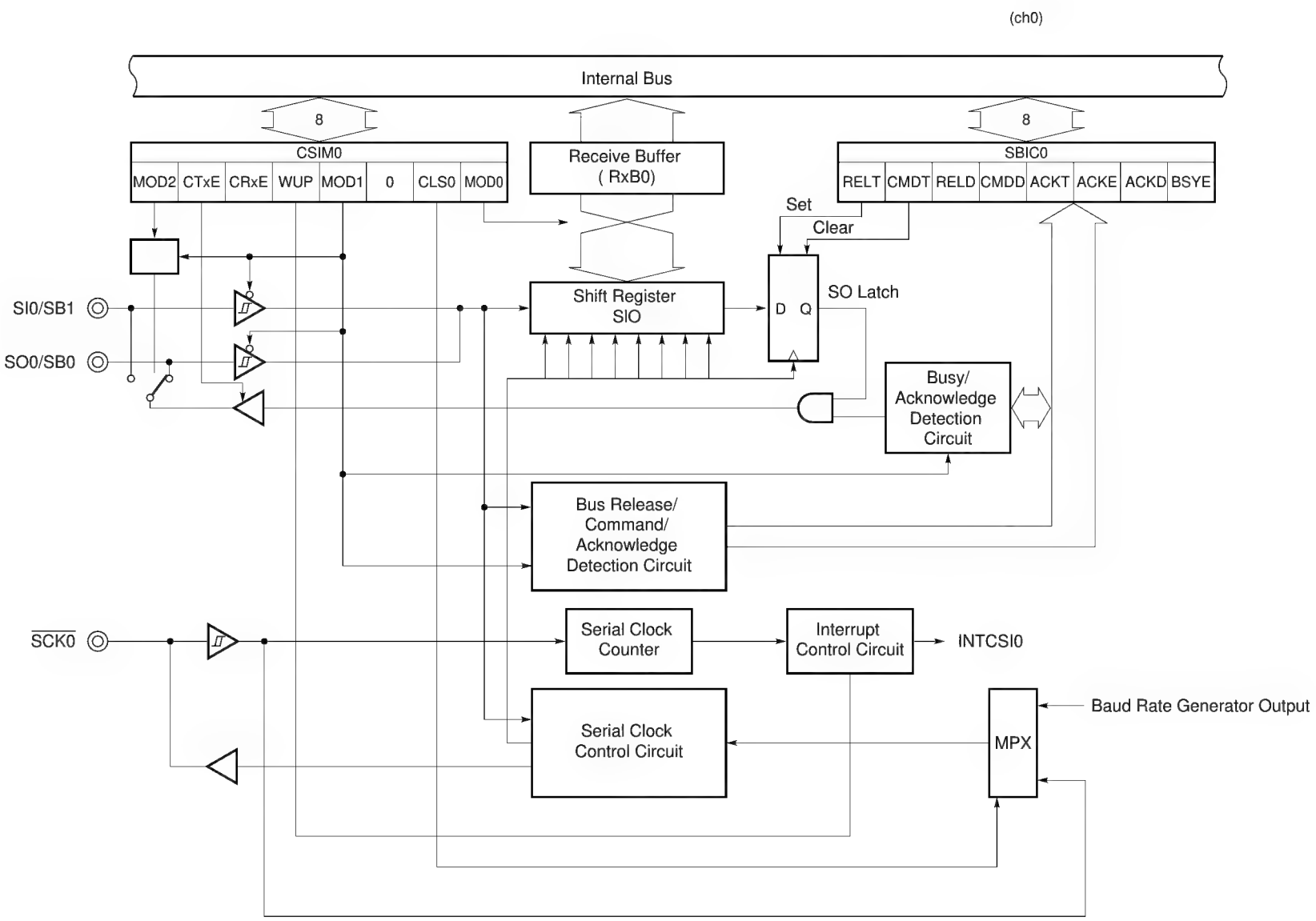
The SI0/SB1 pin functions as the serial data input pin (SI) in the IOE mode, and as the serial data input/output pin SB1 in the SBI mode.

The pins are shown in Table 7-7.

Table 7-7. Clocked Serial Interface Pins

Pin Name	IOE Mode	SBI Mode
SO0/SB0	Serial data output (SO0)	Serial data input/output (SB0)
SI0/SB1	Serial data input (SI0)	Serial data input/output (SB1)
SO1	Serial data output (SO1)	–
SI1	Serial data input (SI1)	–
$\overline{SCK}$	Serial clock input/output	

Figure 7-9. Clocked Serial Interface Block Diagram



The SO, SI and $\overline{SCK}$ pins also function as port 3 pins. When a protocol is set in UART mode, a port is set to the control function (PMCn = 1), the function depends on the clocked serial interface mode register (CSIM) control bits as shown below.

Table 7-8. Function Specification by Control Register (SO/SB0 Pins) ch0

PMC30	PM30	CSIM0		Operating Mode	P30 Read Operation	P30 Write Operation
		MOD1	MOD2			
1	0	1	0	• Serial data input/output (SB0)	Control signal before output buffer is read	Data is written to output latch
	1			• N-ch open drain[Note]	Pin level is read	
	0	1	1	High impedance	Control signal before output buffer is read	
	1				Pin level is read	
	0	0	1/0	• Serial data output (SO)	Control signal before output buffer is read	
	1			• CMOS output	Pin level is read	

Table 7-9. Function Specification by Control Register (SI/SB1 Pins) ch0

PMC31	PM31	CSIM0		Operating Mode	P30 Read Operation	P30 Write Operation
		MOD1	MOD2			
1	0	1	0	High impedance	Control signal before output buffer is read	Data is written to output latch
	1				Pin level is read	
	0	1	1	• Serial data input/output (SB1)	Control signal before output buffer is read	
	1			• N-ch open drain[Note]	Pin level is read	
	0	0	1/0	• Serial data input (SI) • High impedance	Control signal before output buffer is read	
	1				Pin level is read	

Note Connect externally with a pull-up resistor.

Table 7-10. Function Specification by Control Register (SO Pin) ch1

PMC34	PM34	Operating Mode	P34 Read Operation	P34 Write Operation
1	0	• Serial data output (SO) • CMOS output	Control signal before output buffer is read	Data is written to output latch
	1		Pin level is read	

Table 7-11. Function Specification by Control Register (SI Pin) ch1

PMC35	PM35	Operating Mode	P35 Read Operation	P35 Write Operation
1	0	• Serial data input (SI) • High impedance	Control signal before output buffer is read	Data is written to output latch
	1		Pin level is read	

Table 7-12. Function Specification by Control Register ($\overline{SCK}$ Pin)

PMC32/ PMC36	PM32/ PM36	CSIM CLS0	Operating Mode	P32/P36 Read Operation	P32/P36 Write Operation
1	0	0	• External $\overline{SCK}$ input • High impedance	Control signal before output buffer is read	Data is written to output latch
		1	• $\overline{SCK}$ output • CMOS output		
	1	0	• External $\overline{SCK}$ input • High impedance	Pin level is read	
		1	• $\overline{SCK}$ output • CMOS output		

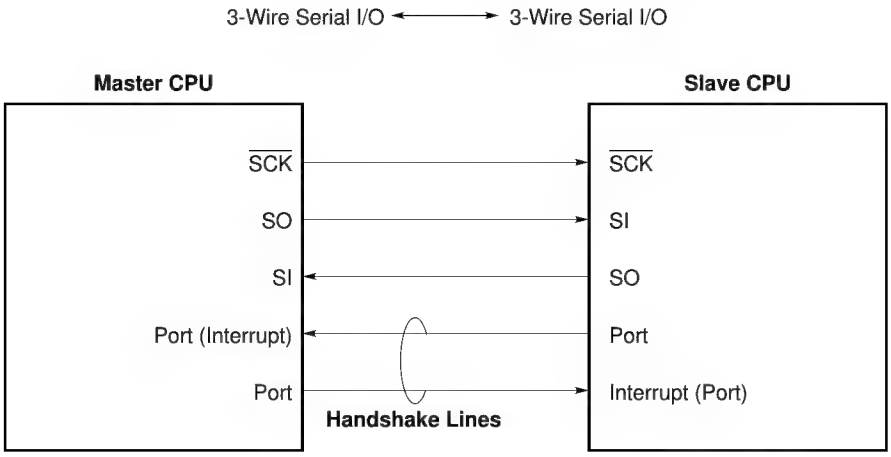
Remark The $\overline{SCK}$ pin is not affected by the setting of the MOD1 and MOD2 bits.

7.5.2 Operation in 3-wire serial I/O mode (IOE mode)

The 3-wire serial I/O mode (IOE mode) is used for communication with devices which incorporate a conventional clocked serial interface.

Communication is basically performed using three lines: the serial clock ($\overline{SCK}$), and serial data output (SO) and serial data input (SI). Handshake lines is not required when multiple devices are connected.

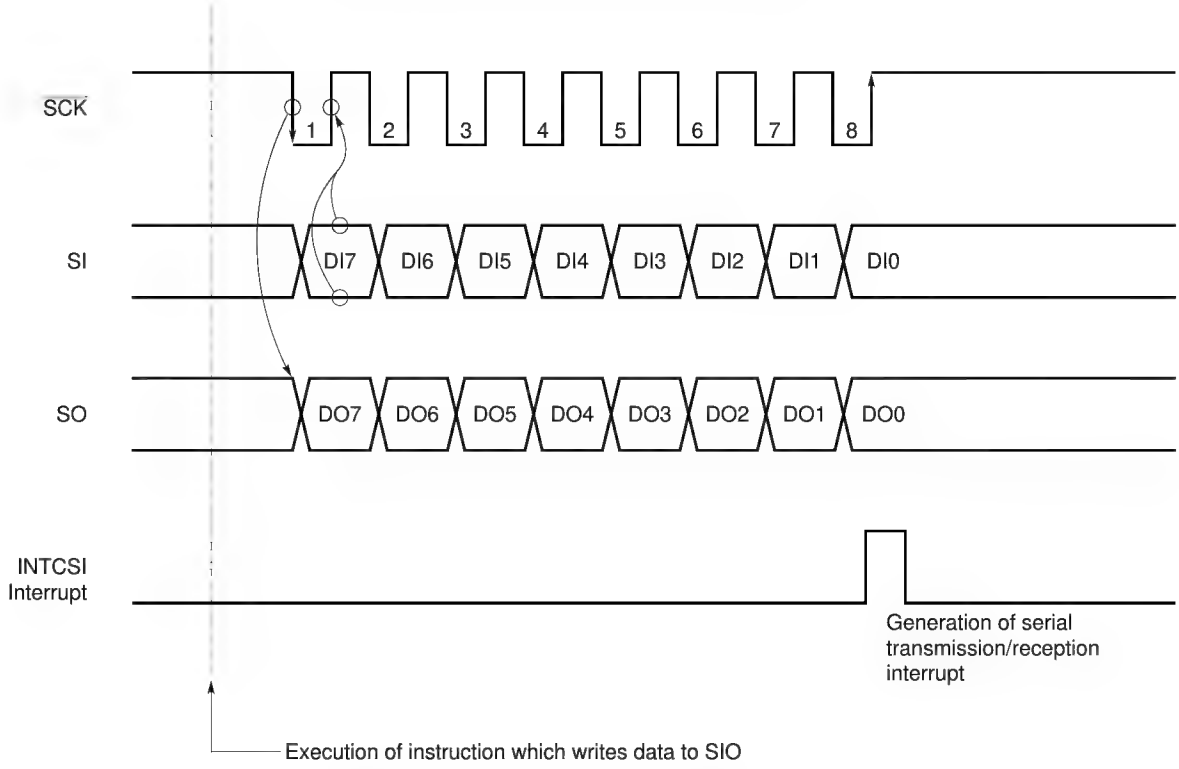
Figure 7-10. Example of IOE Mode System Configuration



The IOE mode transfer timing is shown in Figure 7-11.

The serial transfer transmission/reception functions are controlled by the transmission enable bit (CTxE) and reception enable bit (CRxE) of the clocked serial interface mode register (CSIM), the serial I/O shift register (SIO), and the serial clock ($\overline{SCK}$).

Figure 7-11. IOE Mode Transfer Timing



Remark Transmission : Transmission is synchronized with the falling edge of $\overline{SCK}$ and started
Reception : The receive data is latched at the rising edge of $\overline{SCK}$.

(1) Transmit operations

The SIO register data is synchronized with the falling edge of the $\overline{SCK}$ signal and output in order to the SO pin with MSB at the head.

(a) Enabled state

Setting : Clock serial interface mode register (CSIM) CTxE bit = 1 and CRxE bit = 0

Start-up :
- When the internal clock is selected
  - <1> The transmit enabled state is set.
  - <2> The transmit data is written in the SIO register.
  - <3> The shift clock is output from the $\overline{SCK}$ pin and transmission is started.
- When the external clock is selected
  - <1> The transmit enabled state is set.
  - <2> The transmit data is written in the SIO register.
  - <3> The shift clock is input to the $\overline{SCK}$ pin and transmission is started.

(b) Disabled state

Setting : CSIM register CTxE bit = 0

State of each pin : SO pin is high impedance output.
Data is not output from the SO pin even if the shift clock is input to the $\overline{SCK}$ pin.

(c) Transmit interrupt generation timing

For every 8-bit output of the SIO register contents, serial transmit/receive interrupt (INTCSI) is generated.

(2) Receive operations

Reception is synchronized with the rising edge of the $\overline{SCK}$ signal and the data input to the SI pin is input to the SIO register sequentially. When 8 bits of the data have been received, the receive data in the SIO register is transmitted to the receive buffer (RxB).

(a) Enabled state

Setting : CSIM register CRxE bit = 1, CTxE bit = 0

Start-up :
- When the internal clock is selected
  - <1> The receive enabled state is set.
  - <2> The receive buffer (RxB) is read.
  - <3> The shift clock is output from the $\overline{SCK}$ pin and reception starts.
- When the external clock is selected
  - <1> The receive enabled state is set.
  - <2> The receive buffer (RxB) is read.
  - <3> The shift clock is input to the $\overline{SCK}$ pin and reception starts.

(b) Disabled state

Setting : CSIM register CRxE bit = 0

Operation : Data is not received even though the shift clock is input to the $\overline{SCK}$ pin. However, the data of the receive buffer can be read even in the reception disabled state.

(c) Receive interrupt generation timing

Each time 8 bits of the receive data have been input, serial transmit interrupt (INTCS1) is generated.

(3) Transmit/receive operation

(a) Enable state

Setting : CTxE bit of CSIM register = CRxE bit = 1

Start up :
- When selecting the internal clock
  - <1> Set to transmit/recieve enable state.
  - <2> Write transmit data to the SIO register
  - <3> Transmit/receive operation starts after outputting the shift clock from the $\overline{SCK}$ pin.
- When selecting the external clock
  - <1> Set to transmit/receive enable state.
  - <2> Write transmit data to the SIO register.
  - <3> Transmit/receive operation starts after inputting the shift clock to the $\overline{SCK}$ pin.

(b) Disabled state

Setting : CTxE bit of CSTM register = CRxE bit = 0

Pin state : The SO pin is placed into high-impedance state. The SO pin does not output data even when inputting shift clock to the $\overline{SCK}$ pin.

Data cannot be received. However, receive buffer (RxB) data can be read out.

(c) Transmit/receive interrupt generation timing

Serial transmit/receive interrupt (INTCSI) signal is generated by each 8-bit data transfer.

7.5.3 Serial bus interface mode (SBI mode) operation

In the SBI mode, communication is performed with multiple devices using two lines: the serial clock ($\overline{SCK0}$), and serial bus (SB0 or SB1). An example of an SBI mode system configuration is shown in Figure 7-12.

Serial bus selection is performed by means of the CSIM register. SB0 (SO/SB0) is used in the description given here, but the operation is identical when SB1 (SI/SB1) is selected.

The one device among several which is to be communicated with is selected by outputting an address onto the serial bus (see **Figure 7-13**).

After the communication target has been determined, command and data exchange is performed between the master device and slave device, implementing serial communication (see **Figures 7-14** and **7-15**).

(1) Serial transfer transmission function

In the SBI mode, serial transmission is performed by writing transmit data to the clocked serial I/O shift register (SIO) when the transmission enable bit (CTxE) of the clocked serial interface mode register (CSIM) set (1). After transmission, "0" is written to the shift register.

Shift register shifts are performed and transmit data is held in the SO latch in synchronization with the fall of the serial clock ($\overline{SCK0}$). The data held in the SO latch is output from the SB0 pin MSB-first.

(2) Serial transfer reception function

Serial reception is performed by setting (1) the reception enable bit (CRxE) of the CSIM register, or reading receive data from the receive buffer (RxB) when this bit is set.

Data input to the SB0 pin in synchronization with the serial clock ($\overline{SCK0}$) is latched in the SIO register MSB-first. When specified as the serial bus, SB0 must be pulled high externally with a resistor.

(3) Serial transfer transmit/receive operation

Serial transmission/receive is performed by writing transmit data to the SIO register when CRxE = CTxE = 1. The data on the serial bus remains as it is in the SIO register. A check for a bus collision can be made by comparing the transmit data written to the SIO register with the data input from the serial bus.

Figure 7-12. Example of SBI Mode System Configuration

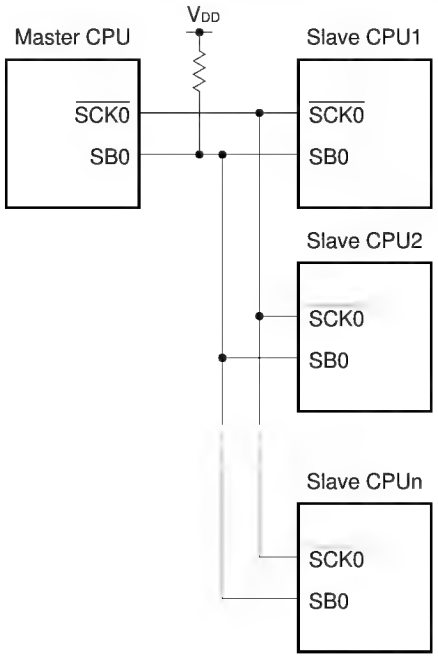


Figure 7-13. Address Transmission from Master Device to Slave Device

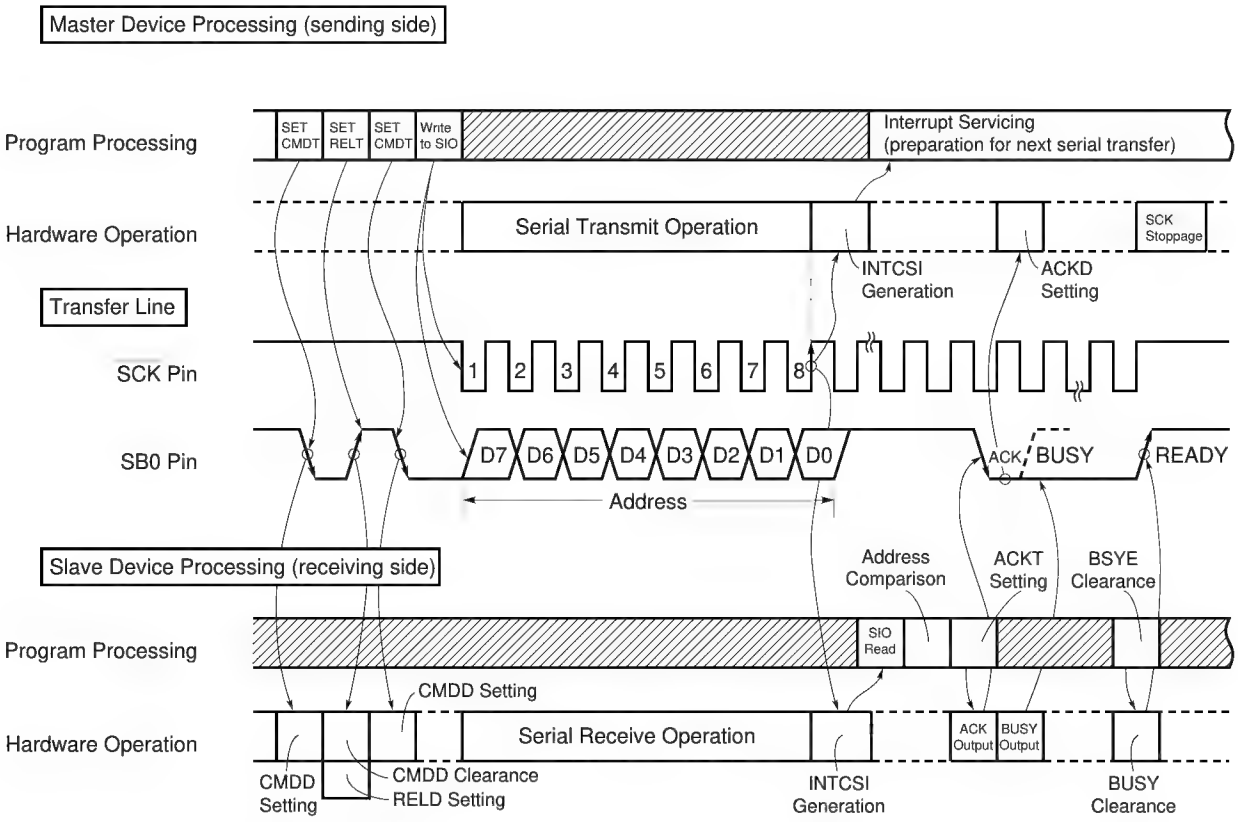


Figure 7-14. Command Transmission from Master Device to Slave Device

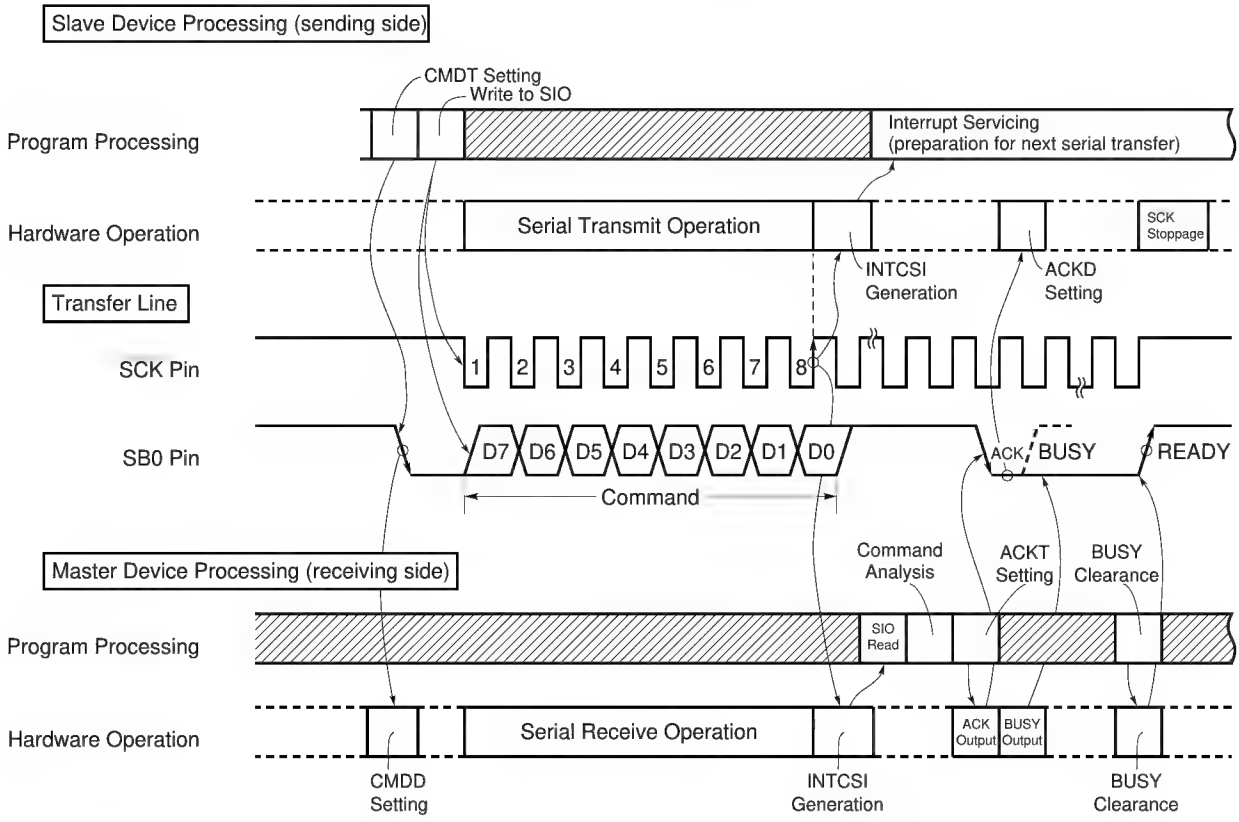


Figure 7-15. Data Transmission from Master Device to Slave Device

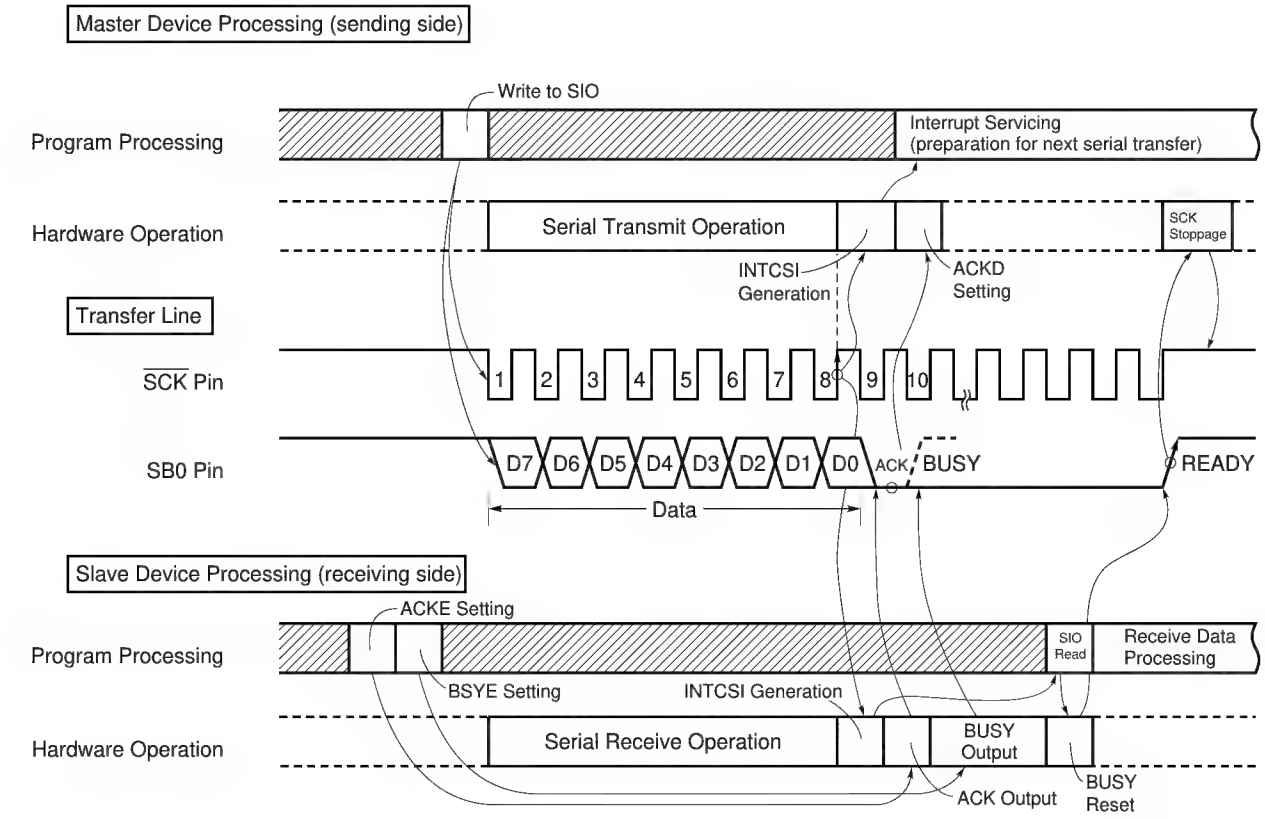
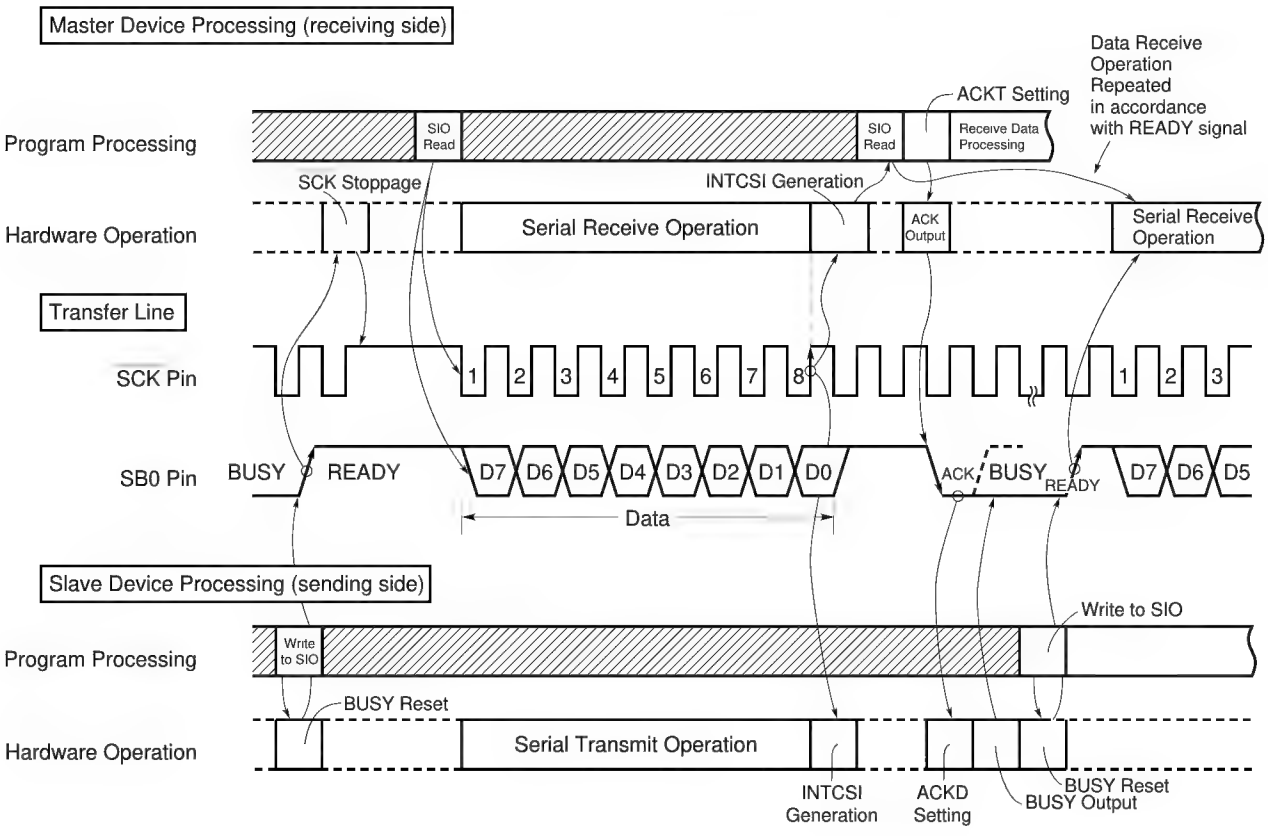


Figure 7-16. Data Transmission from Slave Device to Master Device



(4) Wake-up function

In the SBI mode, a wake-up function is provided which generates an interrupt request signal (INTCSI) only when an address is received.

Wake-up function setting is performed by setting (1) bit 5 (WUP) of the CSIM register.

If the wake-up function is set with serial interface having the system configuration shown in Figure 7-12, a serial transmission/reception interrupt (INTCSI) for all slave devices is generated only when an address is received from the master device. Slave device address match detection is performed by software, and when a matching address is received, the wake-up state is released (WUP = 0) and preparation is made for subsequent command and data transfer. When subsequent command/data transfer is performed, slave devices other than the one which released the wake-up state do not generate a serial transmission/reception interrupt, and thus when transmission/reception is to be performed to/from a particular slave CPU, the other slave CPUs can operate without regard to the serial communication. If there were no wake-up function, an interrupt would be generated each time data is received, and thus all the slave CPUs would be affected by the serial communication. Use of the wake-up function enables processing to be performed efficiently.

The operation and timing of the various signals output from the master device and slave device in the SBI mode are shown in Table 7-13.

Table 7-13. Signals in SBI Mode (1/2)

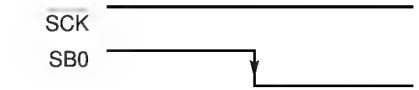
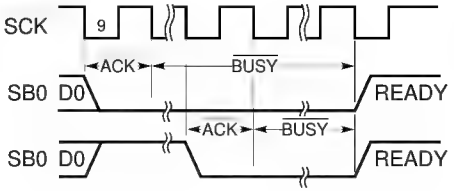
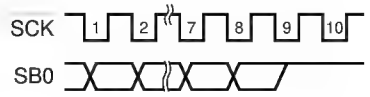
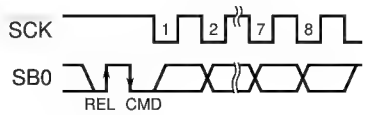
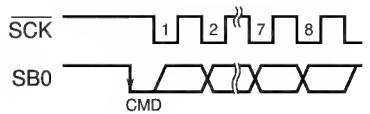
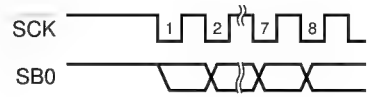
Signal Name	Outputting	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Bus Release signal (REL)	Master	SB0 rising edge when $\overline{SCK}$ = 1		• RELT set	• RELD set • CMDD cleared	Indicates end of transfer processing for slave selected up to now.
Command Master signal (CMD)	Master	SB0 falling edge when $\overline{SCK}$ = 1		• CMDT set	• CMDD set	i) After REL signal output, transmit data is address. ii) No REL signal output. Transmit data is command.
Acknowledge signal ($\overline{ACK}$)	Slave	Low-level signal output to SB0 in 1 $\overline{SCK}$ clock interval after serial reception completion	[Synchronous busy output]	• ACKE=1 • ACKT set	• ACKD set	Reception completed
Busy signal ($\overline{BUSY}$)	Slave	[Synchronous Busy signal:] Low-level signal output to SB0 after Acknowledge signal.		• BSYE=1	—	Serial reception disabled because processing is in progress.
Ready signal (READY)	Slave	High-level signal output to SB0 before start or after completion of serial transfer		• BSYE=0 • Execution of instruction to write data to SIO (transfer start directive)	—	Serial reception enabled

Table 7-13. Signals in SBI Mode (2/2)

Signal Name	Outputting	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Serial Clock ($\overline{SCK}$)	Master	Synchronization clock for output of address/ command/data, ACK signal, synchronous BUSY signal, etc. Address/command/data is transferred in first 8 cycles.		Excution of instruction to write to SIO when CTxE=1 (serial transfer start directive)Note 3	CSIIFNote 1 set (rise of 9th SCK clock cycle) Note 2	Timing of signal output to serial data bus
Address (A0 to A7)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after REL signal and CMD signal output			Note 2	Address value of slave device on serial bus
Command (D0 to D7)	Master	8-bit data transferred in synchronization with $\overline{SCK}$ after CMD signal only is output without output of REL signal			None	Directive message to slave device
Data (D0 to D7)	Master/ slave	8-bit data transferred in synchronization with $\overline{SCK}$ with no output of either REL signal or CMD signal			None	Actual data to be processed by slave or master

Notes 1. Interrupt request flag for serial transmission/reception interrupt (INTCSI).

2. When WUP = 0, CSIIF is always set on the 9th rise of $\overline{SCK}$.
When WUP = 1, CSIIF is set only at the time an address is received.

3. When in the $\overline{BUSY}$ state, the transfer starts after transition to the READY state.

7.5.4 Control registers

(1) Clocked serial interface mode registers (CSIM0, CSIM1)

These are 8-bit registers which specify the basic operating mode of the clocked serial interface.

The CSIM0 register corresponds to ch0, and the CSIM1 register to ch1.

These registers can be read/written from/to by an 8-bit manipulation memory access.

The value of the CSIM registers after a reset is 00H.

	7	6	5	4	3	2	1	0
CSIM1	CTxE	CRxE	0	0	0	MOD0	0	CLS0

	7	6	5	4	3	2	1	0
CSIM0	CTxE	CRxE	WUP	MOD2	MOD1	MOD0	0	CLS0

Operating Clock Specification (ch0, ch1)

CLS0	Clock Specification	SCK Pin
0	External Clock - Slave	Input
1	Internal Clock - Master[Note 1]	Output

Operating Mode Specification (ch0)

MOD1	MOD0	Operating Mode	Transfer Direction
0	0	IOE Mode	MSB First
0	1	IOE Mode	LSB First
1	0	SBI Mode	MSB First
1	1	Setting Prohibited	

Operating Mode Specification (ch1)

MOD0	Operating Mode	Transfer Direction
0	IOE Mode	MSB First
1	IOE Mode	LSB First

Used Pin Specification (ch0)

MOD2	Used Pin & Operation
0	Transmit/receive operation using SO0/SB0 pin
1	Transmit/receive operation using SI0/SB1 pin

Wake-up Operation Specification[Note 2] (ch0)

WUP	Operation
0	Interrupt generated for each transfer
1	Interrupt generated when address is received

Reception Enable/Disable Specification (ch0, ch1)

CRxE	Operation
0	Receive Operation Disabled
1	Receive Operation Enabled

Transmission Enable/Disable Specification (ch0, ch1)

CTxE	Operation
0	Transmit Operation Disabled
1	Transmit Operation Enabled

Notes 1. BRG output

2. Valid in the SBI mode.

Remark The shaded area indicates status after a reset.

MOD1 MOD0 Operating Mode Selection Bits

Specify the serial transfer and protocol operating mode.

[ch0]

MOD1	MOD0	Operating Mode	Transfer Direction	Number of Pin Used	Pins Used
0	0	IOE mode	MSB first	3	SO0/SB0 SI0/SB1 $\overline{SCK0}$
0	1	IOE mode	LSB first		
1	0	SBI mode	MSB first	2	SO0/SB0 (SI0/SB1) $\overline{SCK0}$
1	1	Setting prohibited			

[ch1]

MOD0	Operating Mode	Transfer Direction	Number of Pins Used	Pins Used
0	IOE mode	MSB first	3	SO1 SI1 $\overline{SCK1}$
1	IOE mode	LSB first		

MOD2 Pin Specification Bit (Valid for ch0 only)

Specifies the serial data input/output pins used in the serial bus interface mode (SBI mode).

The same SBI functions can be implemented whether the SO0/SB0 pin or the SI0/SB1 pin is selected as the serial data input/output pin.

When the IOE mode is selected, the MOD2 bit is meaningless.

When the 3-wire IOE mode is selected by MOD0 and MOD1, the MOD2 pin specification is invalid.

WUP Wake-up Operation Specification Bit

Controls the wake-up condition operation. This bit is only valid when the SBI mode has been selected. In the IOE mode, it should be set to "0".

Setting (1) this bit when the master device and a slave device are communicating enables the generation of unnecessary interrupts to be suppressed.

Caution If the WUP bit is set to 1 in the IOE mode, no transfer operations whatsoever will be performed, and no interrupt request (INTCSI) will be generated.

(2) SBI control register (SBIC0)

Used in the SBI mode only.

SBIC0 is an 8-bit register composed of bits which control the serial bus operation and flags which indicate the status. This register can be set/cleared/tested by an 8/1-bit manipulation memory access.

The value of the SBIC0 register after a reset is 00H.

	7	6	5	4	3	2	1	0
SBIC0	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT

Bus Release Trigger Bit

RELT	Processing
0	Bus Line Not Changed
1	Bus Line (SB1/SB0) 0 → 1

Command Trigger Bit

CMDT	Processing
0	Bus Line Not Changed
1	Bus Line (SB1/SB0) 1 → 0

Wake-Up Condition Detection Flag

Command Condition Detection Flag

Acknowledge Trigger Bit

ACKT	Processing
0	ACK Signal Not Output
1	ACK Signal Output

Acknowledge Enable Bit

ACKE	Processing
0	$\overline{\text{ACK}}$ Signal Output Disabled
1	ACK Signal Output Enabled

Acknowledge Detection Flag

ACKD	Operation
0	$\overline{\text{ACK}}$ Signal Not Detected
1	ACK Signal Detected

Busy Enable Bit

BSYE	Processing
0	$\overline{\text{BUSY}}$ Signal Output Disabled
1	$\overline{\text{BUSY}}$ Signal Output Enabled

Caution When using ch0 in the IOE mode, 00H should be written to the SBIC0 register.

Remark The shaded area indicates status after a reset.

RELT Bus Release Trigger Bit (Write-Only)

The serial output latch setting control bit.

This bit is a trigger bit which terminates a series of transfers and places the bus in the idle state.

When this bit is set (1) by software, the serial output latch is set (1).

Setting (1) this bit when the $\overline{SCK0}$ pin is high (when the serial clock is not being output) changes the bus line (SB0 or SB1) from 0 to 1, allowing the serial bus line to be disconnected.

Caution Serial output latch manipulation should be performed while the $\overline{SCK0}$ pin is high (while the serial clock is not being output). If manipulation is performed during serial transmission processing, a correct write operation will not be performed.

CMDT Command Trigger Bit (Write-Only)

The serial output latch clearance control bit.

When this bit is set (1) by software, the serial latch is reset (0).

Setting (1) this bit when the $\overline{SCK0}$ pin is high (when the serial clock is not being output) changes the bus line (SB0 or SB1) from 1 to 0, allowing notification to be made of address/command transmission.

Caution Serial output latch manipulation should be performed while the $\overline{SCK0}$ pin is high (while the serial clock is not being output). If manipulation is performed during serial transmission processing, a correct write operation will not be performed.

Serial latch overwriting conditions are as follows:

Setting conditions
1. Writing "1" to the RELT bit
2. End of transfer in SBI mode

Resetting conditions
1. Writing "1" to the CMDT bit
2. System reset

RELD Wake-up Condition Detection Flag (Read-Only)

RELD is a flag which indicates that the bus line (SB1 or SB0) has changed from 0 to 1 while the $\overline{SCK0}$ pin is high.

When this flag is set (1) and the CMDD flag is set (1), this indicates that the receive data is an address.

Figure 7-17. $\overline{SCK0}$ Pin and Bus Line

Setting condition 0 → 1 transition when $\overline{SCK0}$ is high (when the serial clock is not being output)

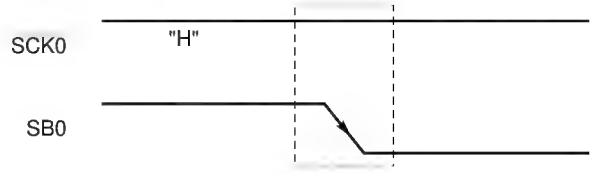
Resetting conditions **1.** When executing a serial start instruction:
Shift register write when CTxE = 1
Shift register read when CTxE = 0*CRxE = 1
CSIM register overwrite when CRxE = 0 → 1
2. System reset

CMDD Command Condition Detection Flag (Read-Only)

CMDD is a flag which indicates that the bus line (SB1 or SB0) has changed from 1 to 0 while the $\overline{SCK0}$ pin is high.

Figure 7-18. $\overline{SCK0}$ Pin and Bus Line (Command Condition Detection)

Setting condition 1 → 0 transition of the bus line when $\overline{SCK0}$ is high



Resetting conditions **1.** When the RELD flag is set
2. Execution of a serial start instruction
3. System reset

ACKT Acknowledge Trigger Bit (Write-Only)

This is a trigger bit which outputs the acknowledge ($\overline{ACK}$) signal.

When "1" is written to this bit after a data transfer, an acknowledge ($\overline{ACK}$) signal is output in synchronization with the next fall of $\overline{SCK0}$.

This bit is used with an operation whereby the acknowledge ($\overline{ACK}$) signal is output after confirming that the receive data or transmit data is correct after a transfer, or after confirming that the receive address and the CPU's own address match.

Caution Once this bit has been set, $\overline{ACK}$ output cannot be canceled.

ACKE Acknowledge Enable Bit

The acknowledge ($\overline{ACK}$) signal output enable bit.

When this bit is set to 1, the acknowledge ($\overline{ACK}$) signal is output in the 9th clock cycle thereafter.

If this bit is changed from 0 to 1 after a data transfer, the $\overline{ACK}$ signal is output in synchronization with the next fall of $\overline{SCK0}$. If the $\overline{ACK}$ signal is to be output constantly, this bit is used in an operation whereby it is set to 1, and the $\overline{ACK}$ signal is output every time. In this case, the bit should be set before transmission/reception is started.

If this bit is set to 0, ACK is not output.

ACKD Acknowledge Detection Flag (Read-Only)

ACKD is a flag which indicates whether the acknowledge ($\overline{ACK}$) signal has been detected.

Setting condition When the acknowledge signal is detected

Clearing conditions **1.** Serial start (actual start)

2. System reset

BSYE Busy Enable Bit

This bit performs $\overline{BUSY}$ signal output control.

If this bit is set (1), the $\overline{BUSY}$ signal is output automatically when the $\overline{ACK}$ signal is detected. If this bit is set before the start of a data transfer, it is possible to drive the SIO line low as the $\overline{BUSY}$ signal after the transfer has ended and one of the devices has output an $\overline{ACK}$ signal.

In addition to being released automatically when a serial start directive is issued, $\overline{BUSY}$ is also released when the BSYE bit is cleared (0) by the program.

Figures 7-19 to 7-23 show the relationship between SBIC0 register bit operation and transfer timing.

Figure 7-19. RELT, CMDT, RELD, and CMDD Operations

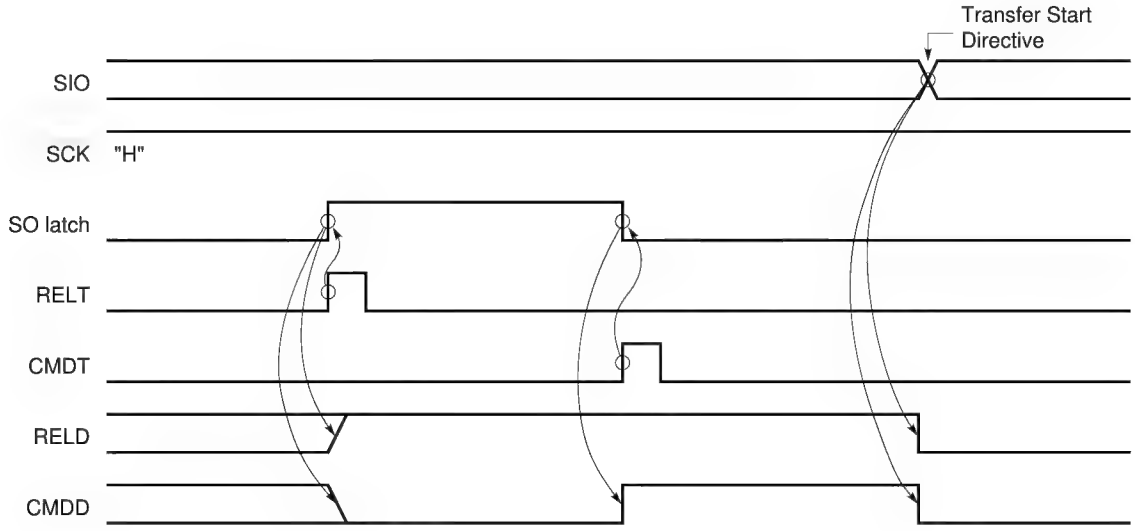


Figure 7-20. ACKT Operation

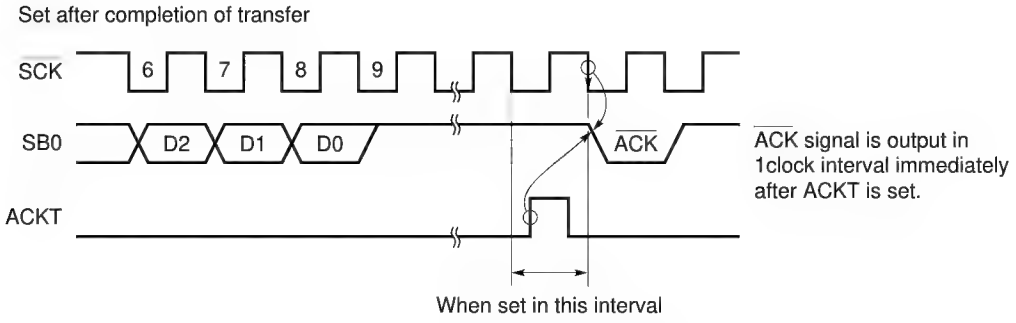
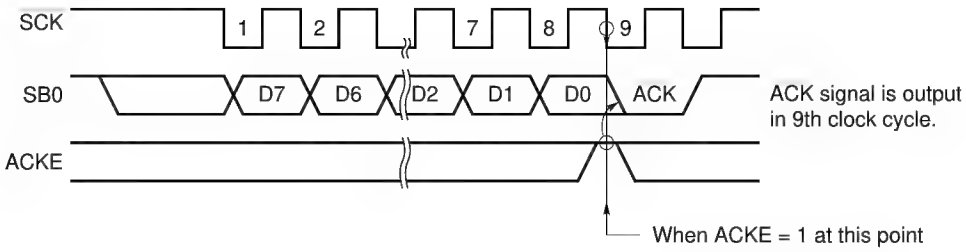
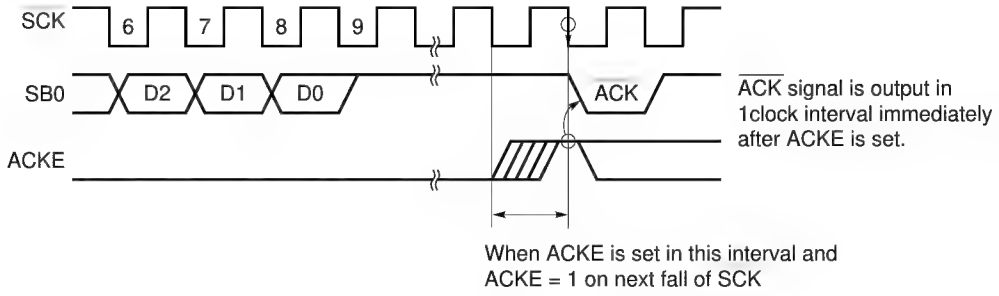


Figure 7-21. ACKE Operation

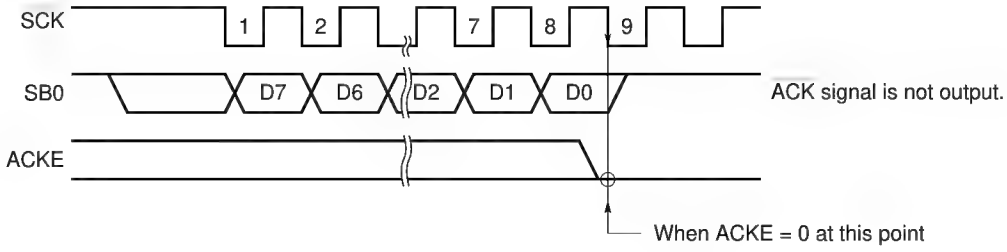
(a) When ACKE = 1 at time of transfer



(b) When set after completion of transfer



(c) When ACKE = 0 on completion of transfer



(d) When ACKE = 1 interval is short

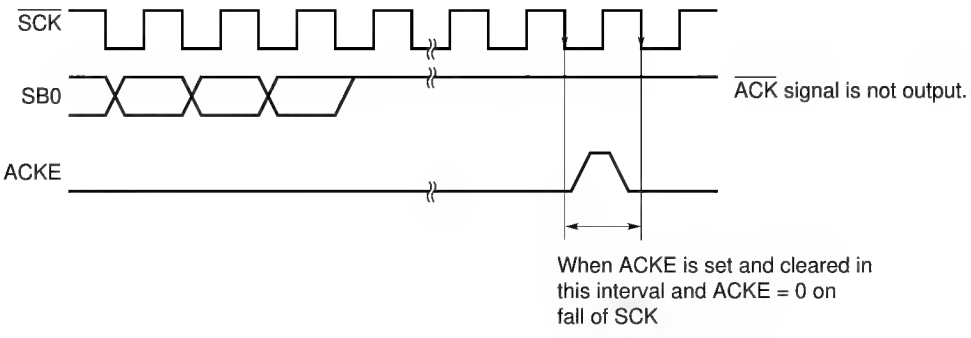
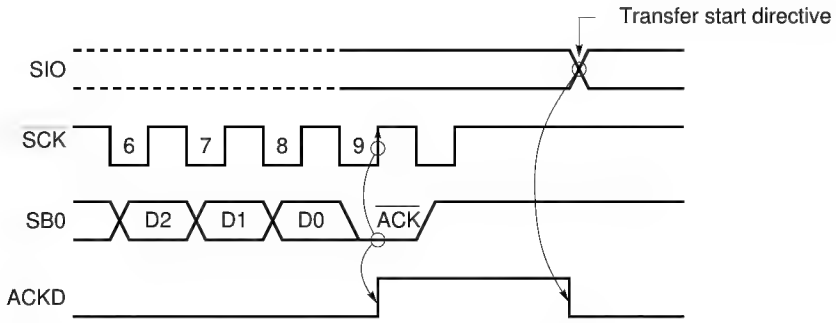
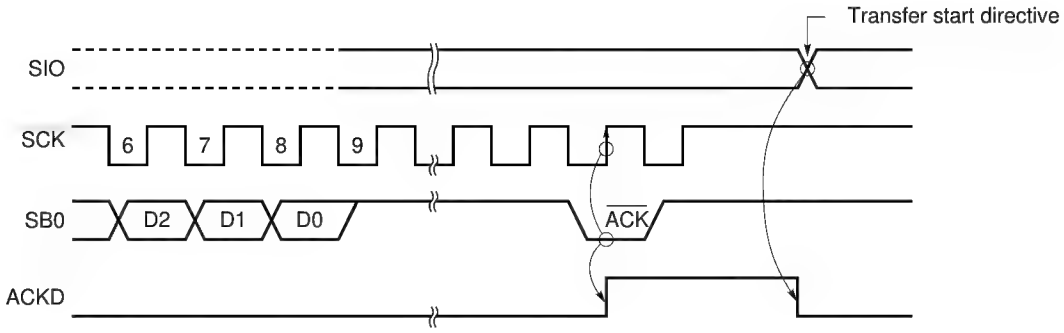


Figure 7-22. ACKD Operation

(a) When $\overline{ACK}$ signal is output in 9th $\overline{SCK}$ clock interval



(b) When $\overline{ACK}$ signal is output after 9th $\overline{SCK}$ clock interval



(c) Clearing timing when transfer start directive is given during BUSY state

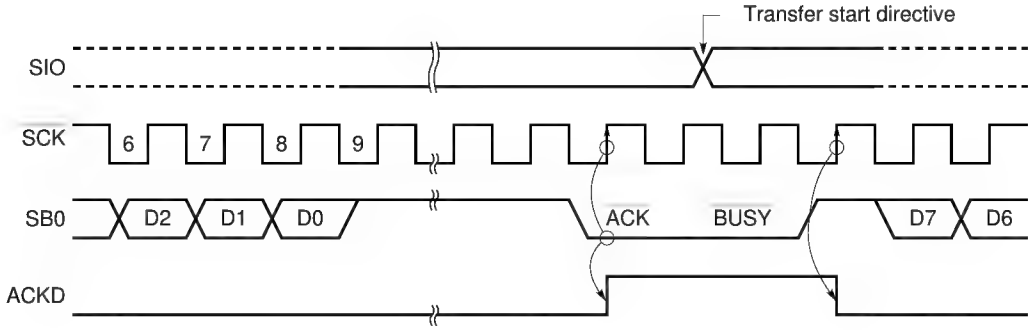
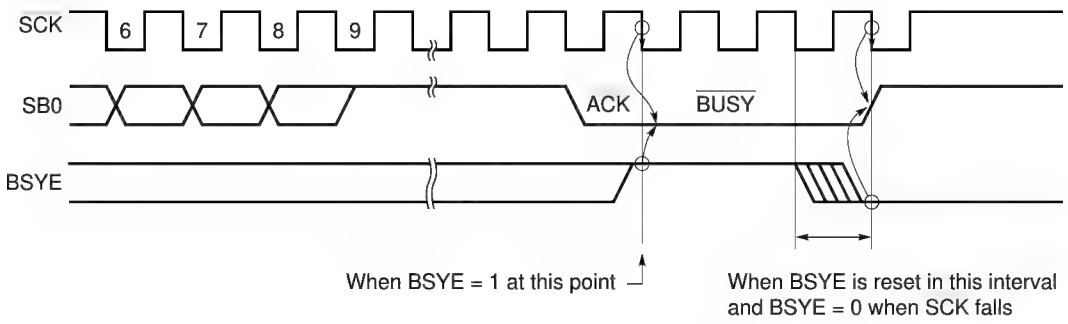


Figure 7-23. BSYE Operation



(3) Clocked serial I/O shift registers (SIO0, SIO1)

These are 8-bit shift registers which perform parallel-serial conversion. Transmit data can only be written from the CPU by an 8-bit manipulation memory access.

An LSB-first or MSB-first shift operation is selected by the MD0 and MD1 bits of the CSIM register.

(4) Receive buffers (RxB0, RxB1)

These are 8-bit registers which hold receive data. They can only be read by an 8-bit manipulation memory access from the CPU.

7.5.5 Interrupt requests

Interrupt requests generated from the clocked serial interface are serial transmission/reception interrupt requests (INTCSI0, INTCSI1).

(1) Serial transmission/reception interrupt requests (INTCSI0, INTCSI1)

In the IOE mode, a serial transmission/reception interrupt request is generated for each serial transfer.

In the SBI mode, the conditions for generation of an interrupt request depend on the wake-up operation specification of the WUP bit in the CSIM register.

If the WUP bit is cleared (0), a serial transmission/reception completion interrupt request is generated after each serial transfer, and if the WUP bit is set (1), an interrupt request is generated when an address is received.

Caution The INTCSI interrupt shares response functions such as interrupt request control register, vector, etc., with the UART INTSR interrupt. The corresponding interrupt control register is selected automatically by protocol selection.

7.5.6 Caution

(1) Setting order in SBI mode

In the SBI mode, the following settings must be made before enabling reception (CSIM register CRxE bit = 1) or enabling transmission (CSIM register CTxE bit = 1):

- Serial clock selection
- SBI mode serial data input/output pin selection

(2) Canceling receive data in IOE mode

If receive operation being performed is stopped in the IOE mode, the CTxE bit and CRxE bit of the CSIM register are both set to 0, the shift counter is reset, and the shift register (SIO) is returned to the initial state.

CHAPTER 8 PARALLEL INTERFACE FUNCTIONS

The V55PI incorporates a parallel interface unit for data input on a Centronics specification interface, and general data input/output.

8.1 Features

- Centronics specification interface compatibility
- Input/output mode switchable by software
- BUSY signal manipulable by software
- BUSY signal and $\overline{ACK}$ signal output timing settable
- Initialization by external interrupt
- Dedicated parallel interface interrupt sources
  - Parallel interface interrupt (INTPAI)
- DMA request signal generation in parallel transmission/reception
  - INTPAI functions as a DMA start trigger.
- Signal pin input characteristic is TTL level (Centronics specification interface)

8.2 Configuration

V55PI incorporates the following control registers (See **8.4 Control Registers** for details).

(1) Parallel interface control register 0 (PAC0)
Controls switching between the input mode and output mode.

(2) Parallel interface control register 1 (PAC1)
Controls the BUSY pin in the input mode.

(3) Parallel interface status register (PAS)

(4) Parallel interface acknowledge interval register (PAI)

(5) Parallel interface buffer (PAB)
Latches input/output data.

Figure 8-1 shows the parallel interface configuration in each mode.

Figure 8-1. Parallel Interface Block Diagram (1/2)

(a) Input mode

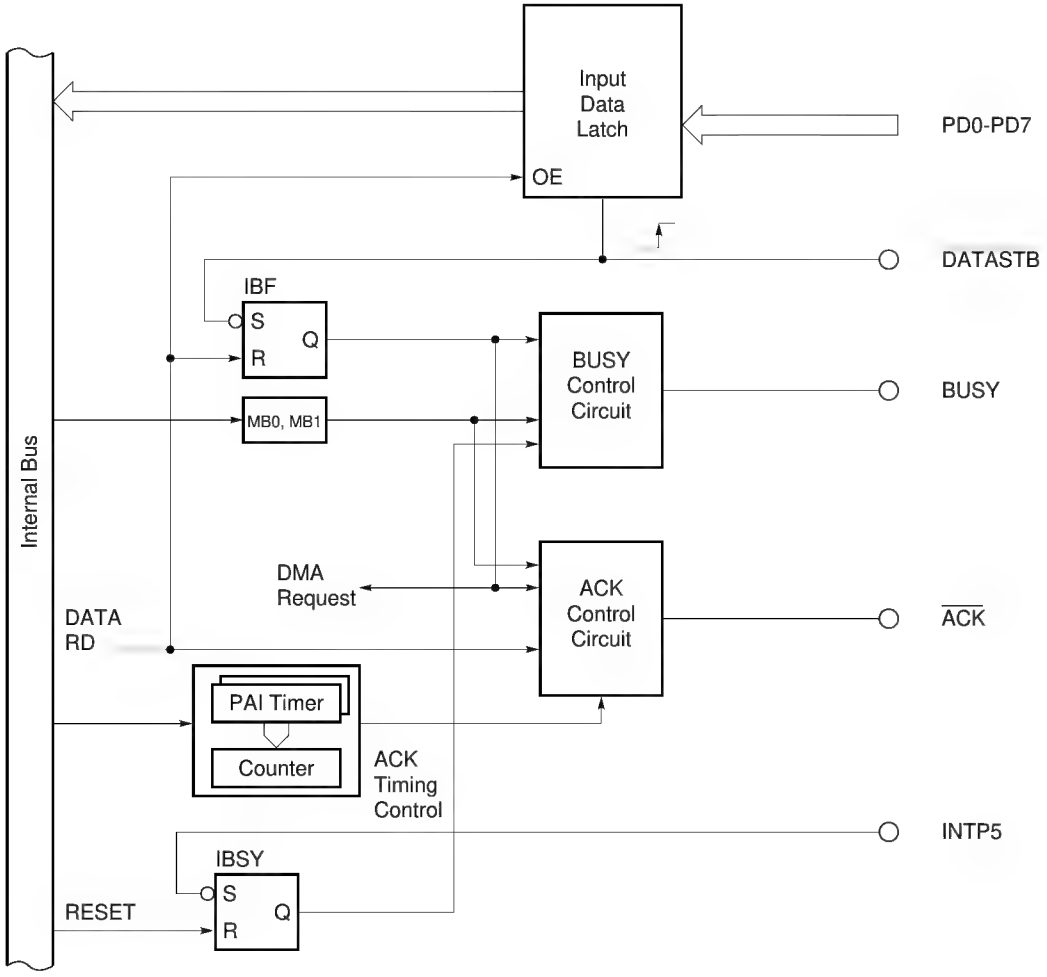
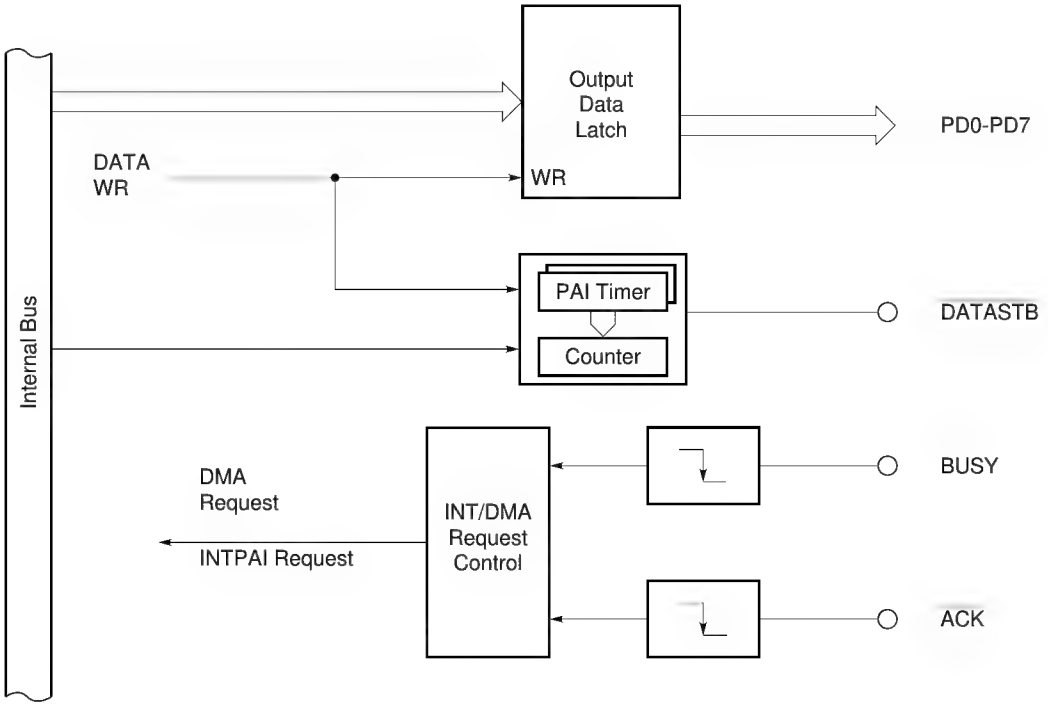


Figure 8-1. Parallel Interface Block Diagram (2/2)

(b) Output mode



8.3 Operation

The interface operations in the input and output modes are described below.

8.3.1 Input mode

The interface operates in input mode when bit 7 (MOD) of the register (PAC0) is set (1).

(1) Basic operations

Data is input as follows.

<1> $\overline{\text{DATASTB}}$ signal input

When the $\overline{\text{DATASTB}}$ signal is input, data of PD0 to PD7 are latched to the PAB register. Bit 7 of the register (PAS) is set (1).

<2> Interrupt request, DMA request generation

The BUSY signal is output when data is latched to the register (PAB), and a DMA request and interrupt request (INTPAI) are generated.

<3> Data reading from PAB register

Data in the register (PAB) is read by the CPU using DMA transfer, macro service, or instruction manipulation. Bit 7 of the register (PAS) is cleared.

<4> Response signal output from $\overline{\text{ACK}}$ pin

The $\overline{\text{ACK}}$ signal is output when data in the register (PAB) is read. At this time, the internal PAI timer is cleared and a count operation starts.

<5> $\overline{\text{ACK}}$ output termination

(2) Transfer timing, control signal output timing

The output timings of BUSY and $\overline{\text{ACK}}$ signals are controlled by the following registers. See the corresponding references for details.

Operation	Register	Bit	Reference
BUSY signal output timing	PAC0	Bit 2 (BSYT)	See **8.4.1**
Data fetch timing to PAB register	PAC0	Bit 3 (STBT)	See **8.4.1**
$\overline{\text{ACK}}$, BUSY signal output	PAC1	Bit 0, 1 (MB0, MB1)	See **Table 8-3**
	PAC0	Bit 0, 1 (ACKT0, ACKT1)	See **Table 8-2**

(3) PAI timer

The internal PAI timer controls the timing of BUSY and $\overline{\text{ACK}}$ signals by continuously comparing the PAI1 and PAI2 registers. Set values corresponding to the active width of the $\overline{\text{ACK}}$ signal to the PAI1 and PAI2 registers (See **8.4.4 Parallel interface acknowledge interval register (PAI1, PAI2)**).

8.3.2 Output mode

The interface operates in the output mode when bit 7 (MOD) of the register (PAC0) is set (0).

(1) Basic operations

Data is output as follows.

<1> Data writing to PAB register

Data is written into the register (PAB) using DMA transfer, macro service, or instruction manipulation.

<2> PAB register data output

Data in the register (PAB) is output from pins of PD0 to PD7. Bit 6 of the register (PAS) is set (1).

<3> $\overline{\text{DATASTB}}$ signal output

The $\overline{\text{DATASTB}}$ signal is output after T1 time elapsed from data output.
The T1 time is specified by the PAI1 register and T2 time ($\overline{\text{DATASTB}}$ signal low level width) is specified by the PAI2 register.

<4> Interrupt request, DMA request generation

An interrupt request (INTPAI) and DMA request are generated when the BUSY or $\overline{\text{ACK}}$ pin is changed from high to low after the $\overline{\text{DATASTB}}$ signal has been output (See **Figure 8-1 (b)**).
Either the BUSY or $\overline{\text{ACK}}$ pin, or both of them can be selected for each request generation timing. Set as shown in Table 8-1 using the port 5 mode control register (PMC5).

Table 8-1. Specification of Trigger Pins for Interrupt Request and DMA Request Generation (Output Mode)

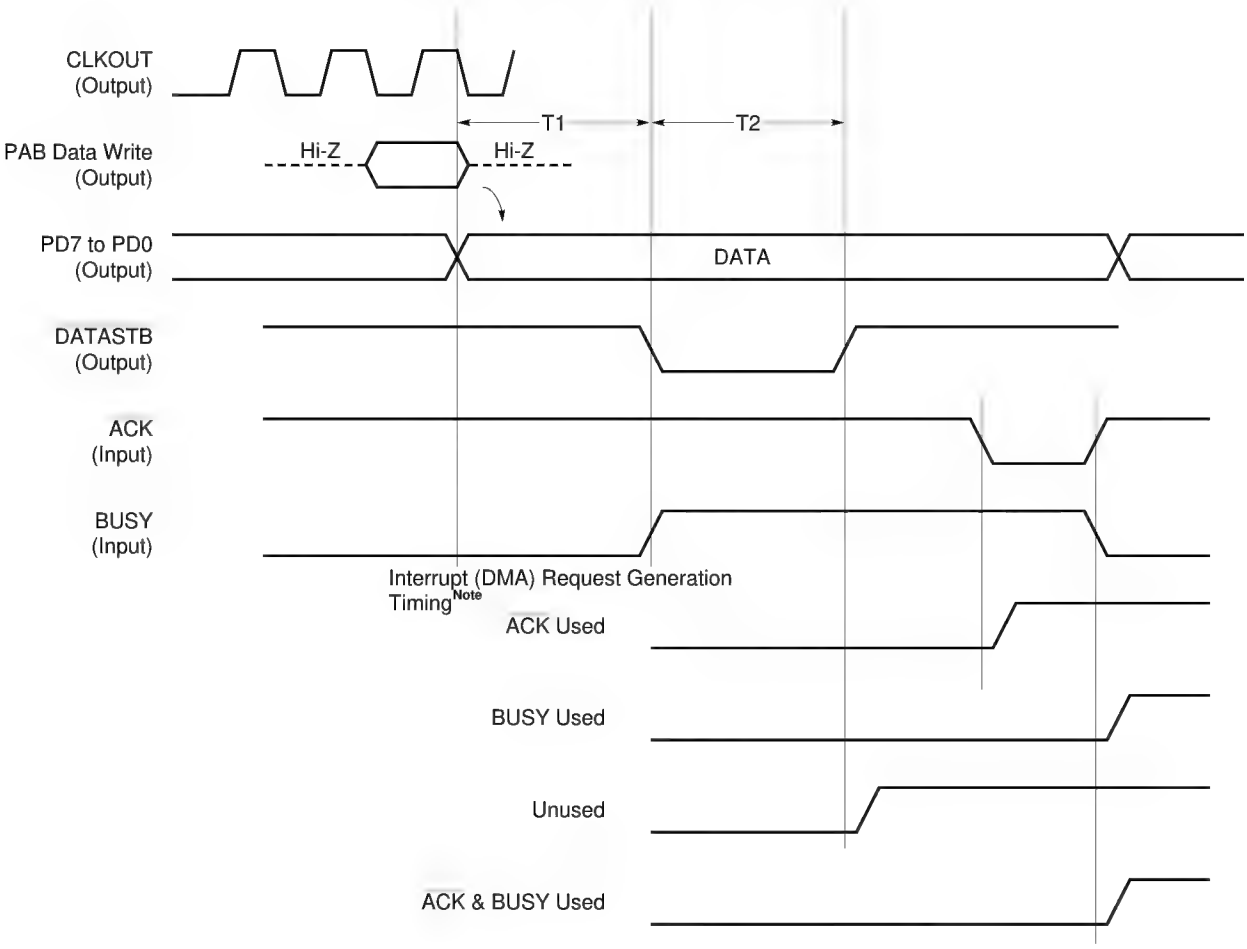
PMC51	PMC52	Contents
1	0	Only $\overline{\text{ACK}}$ pin is used.
0	1	Only BUSY pin is used.
1	1	Both $\overline{\text{ACK}}$ and BUSY pins are used.
0	0	Interrupt request and DMA request are generated immediately after $\overline{\text{DATASTB}}$ signal output which does not use the $\overline{\text{ACK}}$ and BUSY pins.

(2) Differences from port control

If the BUSY pin is set as a dual-function general-purpose output port (PMC52 = 0, PM52 = 0), the P52 (BUSY pin) output level can be specified as desired by instruction manipulation of P52 regardless of the parallel interface operation (PAC0 & PAC1 register settings). In this case, unlike the case in which the BUSY pin is controlled by the PAC1 register, automatic output of the $\overline{\text{ACK}}$ signal, interrupt request, DMA request pending, etc., is not performed.

Figure 8-2 shows the timing in output mode.

Figure 8-2. Timing Chart (Output Mode)



Note Timings vary depending on trigger pins used.

8.4 Control Registers

8.4.1 Parallel interface control register 0 (PAC0)

This is an 8-bit register which specifies input/output mode control, $\overline{\text{DATASTB}}$ signal, BUSY signal and $\overline{\text{ACK}}$ signal output timing control, data fetch timing control, and input data validity/invalidity control. It can be read or written to by an 8/1-bit manipulation memory access.

The value of this register after a reset is 90H.

	7	6	5	4	3	2	1	0
PAC0	MOD	EIN	ACKIN	ACKM	STBT	BSYT	ACKT1	ACKT0

See **Tables 8-2, and 8-3**

BUSY Signal Set Timing Specification[Note]

BSYT	Timing
1	$\overline{\text{DATASTB}}$ Signal Rise
0	$\overline{\text{DATASTB}}$ Signal Fall

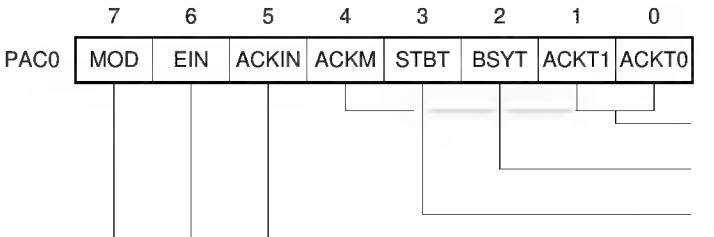
Input Data Fetch Timing Specification[Note]

STBT	Timing
1	$\overline{\text{DATASTB}}$ Signal Rise
0	$\overline{\text{DATASTB}}$ Signal Fall

See next page.

Note Only valid in input mode.

Remark The shaded area indicates status after a reset.



Specification of $\overline{ACK}$ Pin Status in Response to $\overline{DATASTB}$ Input During $\overline{ACK}$ Signal Output[Note]

ACKIN	Status
1	Forcibly Driven High by $\overline{DATASTB}$
0	$\overline{ACK}$ Signal Output Operation Continued

Specification of PAB Register Overwrite Enabling/ Disabling in Receive Mode[Note]

EIN	Operation
1	Overwrite Enabled
0	Overwrite Disabled

Parallel Interface Operating Mode Specification

MOD	Input/Output Mode
0	Output Mode
1	Input Mode

Note Only valid in input mode.

Remark The shaded area indicates status after a reset.

Table 8-2. Specification of $\overline{ACK}$ Signal Output (When MB bit is reset)

ACKM	$\overline{ACK}$ signal output
1	Outputs[Note]
0	Does not output

Note $\overline{ACK}$ signal is not output in the following cases.

- Unread reception data remains in the PAB register (See **Figures 8-13** to **8-15**).
- MB1 and MB0 bits are reset while the $\overline{ACK}$ signal is automatically output (See **Figures 8-22** to **8-24**). The $\overline{ACK}$ signal being output has higher priority.

Table 8-3. Setting of ACKM, ACKT1, ACKT0 and Timing of BUSY Signal and $\overline{ACK}$ Signal

ACKM	ACKT1	ACKT0	BUSY signal reset (L) timing		$\overline{ACK}$ signal output timing	
			After PAB register data read	After MB1 and MB0 bits reset	After PAB register data read	After MB1 and MB0 bits reset
1	0	0	Reset after T1 elapsed		Active immediately after, Inactive after T1+T2 elapsed	
	0	1	Reset immediately after		Active immediately after, Inactive after T1 elapsed	
	1	0	Reset after T1 elapsed			
	1	1	Setting disabled			
0	0	0	Reset after T1 elapsed	Reset immediately after	$\overline{ACK}$ signal is not output	
	0	1	Reset immediately after			
	1	0	Reset after T1 elapsed			
	1	1	Setting disabled			

Remark The periods T1 and T2 are set by the parallel interface acknowledge interval register (PAI1, PAI2).

MOD Parallel Interface Operating Mode Specification

Specified as "1" when the parallel interface is used in input mode, and as "0" when used in output mode. The function of the other flags and the parallel interface buffer (PAB) depends on the setting of this bit. When changing the set value, the data in parallel interface control register 1 (PAC1) must also be re-set.
When the MOD bit is changed, the unit operation is suspended and initialization performed as follows:
It takes 1.5 clock cycles from the point at which the MOD bit is changed until the internal circuit is switched.

- The contents of the parallel interface status register (PAS) are initialized.
- Operation of the PAI timer is stopped (T1 & T2 count).
- The contents of the parallel interface buffer (PAB) are undefined.
- Output pin signals ($\overline{ACK}$, $\overline{DATASTB}$) return to the inactive state. The BUSY signal is driven high (PAC0 & PAC1 and PAI1 & PAI2 do not change).

EIN Specification of PAB Register Overwrite Enabling/Disabling in Receive Mode

Only valid in the input mode. In the output mode, "0" should be written to this bit.
Specifies the operation of the PAB register when the write data strobe signal ($\overline{DATASTB}$) is input when the buffer full flag (IBF) in the parallel interface status register (PAS) is set (1) (when there is unfetched receive data in the PAB register). This bit does not affect BUSY output signal or interrupt/DMA request signal generation.

EIN = 0: PAB register overwrite disabled mode
If previous receive data remains unfetched in the PAB register, the receive data is retained and attempted data writes are invalid until the data is read by an instruction (DMA), etc.,

EIN = 1: PAB register overwrite enabled mode
Even if previous receive data remains unfetched in the PAB register, the PAB register contents are updated if the $\overline{DATASTB}$ is input.

When the buffer full flag (IBF) in the parallel interface status register (PAS) is reset (0), data is written to the PAB register by $\overline{DATASTB}$ signal input irrespective of the value of the EIN bit.

ACKIN Specification of $\overline{ACK}$ Pin Status in Response to $\overline{DATASTB}$ Signal Input during $\overline{ACK}$ Signal Output

Only valid in the input mode. In the output mode, "0" should be written to this bit.

ACKIN = 1: If the $\overline{DATASTB}$ signal is input during an $\overline{ACK}$ signal output operation, the $\overline{ACK}$ pin is forcibly returned to the high level (inactive). At the same time, the internal PAI timer T1 and T2 counts are stopped and terminated (this is the mode in which $\overline{ACK}$ signal output and the T1 & T2 counts are forcibly discontinued).

ACKIN = 0: The $\overline{ACK}$ signal output operation continues despite $\overline{DATASTB}$ signal input, and the operation of the internal PAI timer is not affected. However, data is stored in the PAB register (see **Figures 8-28** and **8-29**).

ACKM **Specification of $\overline{\text{ACK}}$ Output when MB Bit is Reset**

Only valid in the input mode. In the output mode, "0" should be written to this bit.

(1) Specification of $\overline{\text{ACK}}$ output when MB bit is reset

The $\overline{\text{ACK}}$ signal is automatically output once each time receive data is fetched from the PAB register by instruction manipulation, DMA, and macro service ($\overline{\text{ACK}}$ signal automatic output). It is not affected by the ACKM bit.

The ACKM bit also specifies whether $\overline{\text{ACK}}$ signal is output when both MB1 and MB0 bits of the PAC1 register are reset (0) (See **Table 8-2**, **Figures 8-10** to **8-12**).

If ACKM is set to 1, when initialization is performed the BUSY pin falls from high to low, and it is then possible to output the $\overline{\text{ACK}}$ signal, etc.

(2) Specification of BUSY signal reset timing

The reset timing of the BUSY signal varies according to the ACKM bit settings (See **Table 8-3**).

STBT **Input Data Fetch Timing Specification**

Only valid in the input mode. In the output mode, "0" should be written to this bit.

This bit specifies whether a fetch of input data into the PAB register is to be performed on the rising or falling edge of the $\overline{\text{DATASTB}}$ signal (see **Figures 8-3** to **8-6**).

BSYT **BUSY Signal Set Timing Specification**

Only valid in the input mode. In the output mode, "0" should be written to this bit.

This bit specifies whether the timing for automatic output of a high-level signal from the BUSY pin when the $\overline{\text{DATASTB}}$ signal is input is to be synchronized with the rise or fall of the $\overline{\text{DATASTB}}$ signal.

The BUSY signal set timing can be set independently of the PAB register input data fetch timing specified by the STBT bit (See **Figures 8-3** to **8-6**).

ACKT1, **ACKT0** **BUSY Signal Reset Timing & $\overline{\text{ACK}}$ Signal Output Timing Specification**

Only valid in the input mode. In the output mode, "0" should be written to these bits. These bits specifies the output timing of the $\overline{\text{ACK}}$ signal. The reset timing of the BUSY signal is specified by setting these bits and the ACKM bits (See **Table 8-3**).

8.4.2 Parallel interface control register 1 (PAC1)

This is an 8-bit register which performs BUSY pin control.

This register can be read or written to by an 8/1-bit manipulation memory access.

The value of this register after a reset is 03H.

	7	6	5	4	3	2	1	0
PAC1	0	0	0	0	0	INIT	MB1	MB0

BUSY Pin Manipulation Specification[Note1]

MB1	MB0	Status
0	0	BUSY Pin Driven Low (When a reception data is in the PAB register: High level immediately after setting. When after a certain time[Note2] has elapsed: Low level.)
0	1	Setting Prohibited
1	0	Sets the BUSY pin to high level after a certain time[Note3] has elapsed (low level immediately after setting).
1	1	BUSY Pin is Forcibly Driven High

Specification of BUSY Pin Operation in Case of INTP5 Input[Note1]

INIT	Operation
1	BUSY Pin is Forcibly Driven High in Synchronization with INTP5 Input.
0	Operation does not Change

Notes 1. Only valid in input mode.

2. A certain time: This time varies according to the setting of the ACKT1 and ACKT0 bits of the PAC0 register.

- ACKT1=0, ACKT0=1:
  Elapsed time until a reception data is completely read out from the PAB register.
- ACKT1=0, ACKT0=0, or ACKT1=1, ACKT0=0:
  Elapsed time until a reception data is completely read out from the PAB register+elapsed time (T1) specified by the PAI1 register after a data is read out from the PAB register.

3. A certain time: This time is when ACKM bit of the PAC0 register is 1 and varies according to the setting of ACKT1 and ACKT0 bits.

- ACKT1=0, ACKT0=1:
  Elapsed time (T1) specified by the PAI1 register after a data is read out from the PAB register.
- ACKT1=0, ACKT0=0, or ACKT1=1, ACKT0=0:
  Elapsed time (T2) specified by the PAI2 register.

Remark The shaded area indicates status after a reset.

MB1, MB0 BUSY Pin Manipulation Specification

Only valid in the input mode. In the output mode, "00" should be written to these bits.
Bit for setting the BUSY pin to high or low level.
Operations when $\overline{\text{DATASTB}}$ signal is input is described below.

- **MB1=0, MB0=0:**
  Input data is stored in the PAB register.
  After that, the BUSY pin becomes automatically high at the timing specified by the BSYT bit of the PAC0 register and generates an interrupt request (INTPAI) and DMA request.

- **MB1=1, MB0=0, or MB1=1, MB0=1:**
  Input data is stored in the PAB register.
  The interrupt request (INTPAI) and DMA request are held pending until MB1=0/MB0=0 is set.

INIT Specification of BUSY Pin Operation in case of INTP5 Input

Only valid in the input mode. In the output mode, "0" should be written to this bit.
This bit enables/disables automatic transition of the BUSY pin to the high level when an INTP5 external interrupt is input.

INIT=0: The operation of this unit does not change even if there is INTP5 input. INTP5 functions simply as an external interrupt input pin.
INIT=1: The BUSY pin is forcibly driven high in synchronization with INTP5 input (valid edge input) Even if there is INTP5 input, the status of the MB1 & MB0 bits and the PAB register is not affected. Also, $\overline{\text{ACK}}$ signal output and the PAI timer count (T1 & T2 counts) continue to be executed.

The fact that the BUSY pin has been driven high by INTP5 input can be ascertained through the generation of an INTP5 interrupt and by reading the IBSY flag in the parallel interface status register (PAS).

8.4.3 Parallel interface status register (PAS)

This register has allocated to it status flags which indicate the internal status of the parallel interface unit.

This register can be written to or read by an 8-bit memory access.

After a reset or input/output mode switchover (manipulation of the MOD bit in the PAC0 register), this register has a value of 40H.

	7	6	5	4	3	2	1	0
PAS	IBF	OBE	0	0	IBSY	0	0	ERR

Error Flag[Note1]

ERR	Status
1	Set by DATASTB Input when IBF = 1
0	Reset by DATASTB Input when IBF = 0

BUSY Pin Status After INTP Input[Note1]

IBSY	BUSY Pin Status
1	BUSY Pin Driven High by INTP5 Input
0	BUSY Pin Low

Buffer Empty Flag[Note2]

OBE	PAB Register Status
1	Data has been Transmitted
0	Data has been Written

Buffer Full Flag[Note1]

IBF	PAB Register Status
1	Data has been Received
0	No Data

Notes 1. Valid in the input mode only.

2. Valid in the output mode only.

Remark The shaded area indicates status after a reset

IBF Buffer Full Flag

IBF is valid in the input mode only, and is undefined in the output mode.

This flag indicates whether or not there is receive data in the parallel interface buffer (PAB). It is set (1) when data is input to the PAB register by the $\overline{\text{DATASTB}}$ signal, and reset (0) when the PAB register is read.

OBE Buffer Empty Flag

OBE is valid in the output mode only, and is undefined in the input mode.

This flag indicates whether or not PAB register transmit data has been output from ports PD0 through PD7. It is reset (0) when output data is written to the PAB register by an instruction, and set (1) when PAB register data is output to ports PD0 through PD7 and the PAB register is empty.

IBSY Busy Flag

IBSY is valid in the input mode only, and is undefined in the output mode.

This is a status flag which indicates that the BUSY pin has been driven high by INTP5 external interrupt pin input (valid edge input). When INIT is set to 1 in control register 1 (PAC1), the IBSY flag is set (1) in synchronization with interrupt signal input on the INTP5 external interrupt pin, and the BUSY pin is simultaneously driven high.

This flag can be reset (0) by executing an instruction which writes "1" to it. The flag contents are not changed by execution of an instruction which writes "0".

If this flag is not reset (0), the BUSY pin will not return to the low level. When this flag is set (1), the BUSY pin is not driven low even if receive data is read or the MB1 & MB0 bits in control register 1 (PAC1) are reset (00).

ERR Error Flag

ERR is valid in the input mode only, and is undefined in the output mode.

This is a status flag which is set (1) if the $\overline{\text{DATASTB}}$ signal is input before receive data has been read from the PAB register (when IBF = 1). The above operation is performed irrespective of the specification of the EIN bit in control register 0 (PAC0).

This flag is automatically reset (0) by $\overline{\text{DATASTB}}$ signal input when IBF = 0.

ERR can be reset (0) by executing an instruction which writes "1" to it. The flag contents are not changed by execution of an instruction which writes "0".

8.4.4 Parallel interface acknowledge interval register (PAI1, PAI2)

Only register writing can be performed by 8-bit manipulation memory accesses.

The PAI1 and PAI2 register values are undefined after reset.

Specification subjects differ according to input mode and output mode.

(1) Input mode

The active width of the $\overline{ACK}$ signal is specified as follows. The active width is expressed in T1 or T1+T2 using a ACKT bit value of the PAC0 register. The internal PAI timer count time T1 and T2 are specified by the PAI1 and PAI2 registers respectively.

- **ACKT0=0, ACKT1=1, or ACKT0=1, ACKT1=0:**
  T1=2/ϕ × PAI1 set value

- **ACKT0=0, ACKT1=0:**
  T1+T2=(2/ϕ × PAI1 set value) + (2/ϕ × PAI2 set value)

Relations between typical PAI set values and $\overline{ACK}$ output widths when ϕ =12.5 MHz and 16 MHz are shown in Table 8-4.

Table 8-4. Typical PAI Set Values and $\overline{ACK}$ Output Width

(1) ϕ = 12.5 MHz (Unit: μs)

PAIn	$\overline{ACK}$ Output Width
0	40.96
1	0.16
3	0.48
6	0.96
13	2.08
25	4.0
31	4.96
50	8.0
125	20.0

(n=1, 2)

(2) ϕ = 16 MHz (Unit: μs)

PAIn	$\overline{ACK}$ Output Width
0	32.0
1	0.125
4	0.5
8	1.0
16	2.0
32	4.0
40	5.0
64	8.0
128	16.0

(n=1, 2)

(2) Output mode

The output delay time (T1) of the $\overline{DATASTB}$ signal and the active width (T2) of the $\overline{DATASTB}$ signal are specified as follows. The internal PAI timer count time T1 and T2 are specified by the PAI1 and PAI2 registers respectively.

T1=2/ϕ × PAI1 set value
T2=2/ϕ × PAI2 set value

8.4.5 Parallel interface buffer register (PAB)

PAB can be read/written to by an 8-bit memory manipulation instruction.

(1) In input mode

This register can be read only, and cannot be written to.

It functions as an input buffer register which latches the data on data input/output pins PD0 through PD7 in synchronization with the $\overline{\text{DATASTB}}$ signal input.

The IBF flag is set (1) when data is written to this register, and reset (0) when data is read.

(2) In output mode

This register can be written to only, and is undefined if read.

It functions as an output buffer register which latches the data to be output to data output pins PD0 through PD7 in synchronization with the $\overline{\text{DATASTB}}$ signal.

The OBE flag is reset (0) when data is written to this register by an instruction, and set (1) when data is output from this register to PD0 through PD7.

8.5 Interrupt Requests

8.5.1 In input mode

An INTPAI interrupt request signal and DMA request signal are generated when data is fetched into the parallel interface buffer (PAB) by $\overline{\text{DATASTB}}$ signal input.

However, if the MB1 & MB0 bits = 1× and the BUSY pin is high when data is fetched into the PAB register, INTPAI interrupt request signal and DMA request signal generation is held pending until the MB1 & MB0 bits are reset (1× → 00).

8.5.2 In output mode

An INTPAI interrupt request signal and DMA request signal are generated when a valid response signal is input from off-chip to the BUSY pin and $\overline{\text{ACK}}$ pin after data written in the PAB register is output together with the $\overline{\text{DATASTB}}$ signal.

The same timing is used for INTPAI interrupt request signal generation and DMA request signal generation.

8.6 Operation Timing

Parallel interface unit operation timing are shown below.

Figure No.	Mode	ACKT1, 0	ACKM	Operation Condition: MB1 & MB0 Bit Setting Condition etc.
Figure 8-3	Input	00	–	Timing specification for BUSY signal output and DMA transfer request &
Figure 8-4	Input	00	–	interrupt request generation according to STBT & BSYT bit setting
Figure 8-5	Input	00	–	
Figure 8-6	Input	00	–	
Figure 8-7	Input	00	–	$\overline{ACK}$ signal output according to ACKT1 & ACKT0 bit specification
Figure 8-8	Input	01	–	
Figure 8-9	Input	10	–	
Figure 8-10	Input	00	1/0	$\overline{ACK}$ signal output due to MB1 & MB0 bit reset
Figure 8-11	Input	01	1/0	MB1, MB0: 1× → 00
Figure 8-12	Input	10	1/0	
Figure 8-13	Input	00	–	Interrupt (DMA) request hold operation when $\overline{DATASTB}$ signal is input after
Figure 8-14	Input	01	–	MB1 & MB0 bit setting
Figure 8-15	Input	10	–	
Figure 8-16	Input	00	–	
Figure 8-17	Input	01	–	
Figure 8-18	Input	10	–	
Figure 8-19	Input	00	–	When $\overline{DATASTB}$ signal is input before MB1 & MB0 bit setting
Figure 8-20	Input	01	–	
Figure 8-21	Input	10	–	
Figure 8-22	Input	00	–	
Figure 8-23	Input	01	–	
Figure 8-24	Input	10	–	
Figure 8-25	Input	00	1/0	$\overline{ACK}$ signal output due to MB1 & MB0 bit reset
Figure 8-26	Input	01	1/0	When $\overline{DATASTB}$ signal is input before MB1 & MB0 bit setting
Figure 8-27	Input	10	1/0	
Figure 8-28	Input	00	0	When $\overline{DATASTB}$ signal is input continuously (ACKIN=0)
Figure 8-29	Input	01	0	
Figure 8-30	Input	10	0	
Figure 8-31	Input	00	0	When $\overline{DATASTB}$ signal is input continuously (ACKIN=0)
Figure 8-32	Input	01	0	Contention with MB1 & MB0 specification
Figure 8-33	Input	10	0	

(BUSY pin ▽, PAB ▼)

Figure 8-3. Timing Chart (STBT=0, BSYT=0)

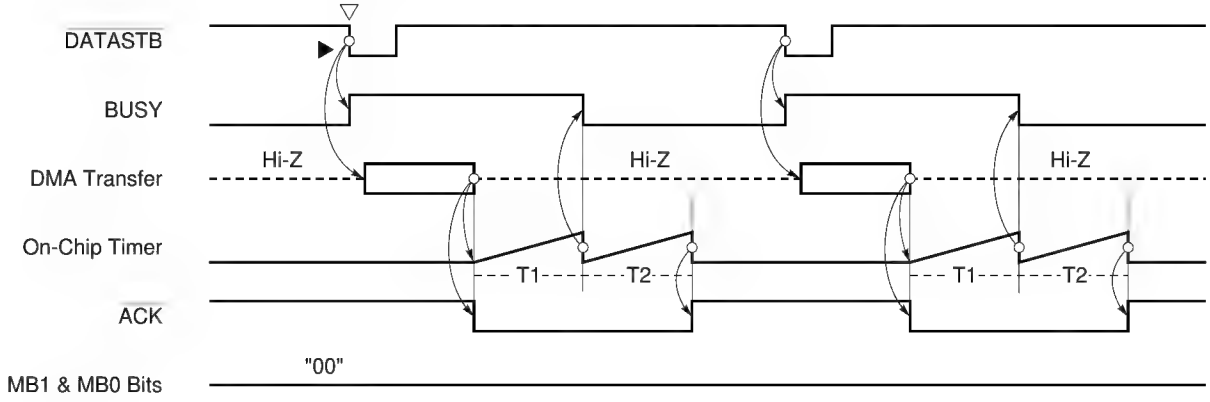


Figure 8-4. Timing Chart (STBT=1, BSYT=0)

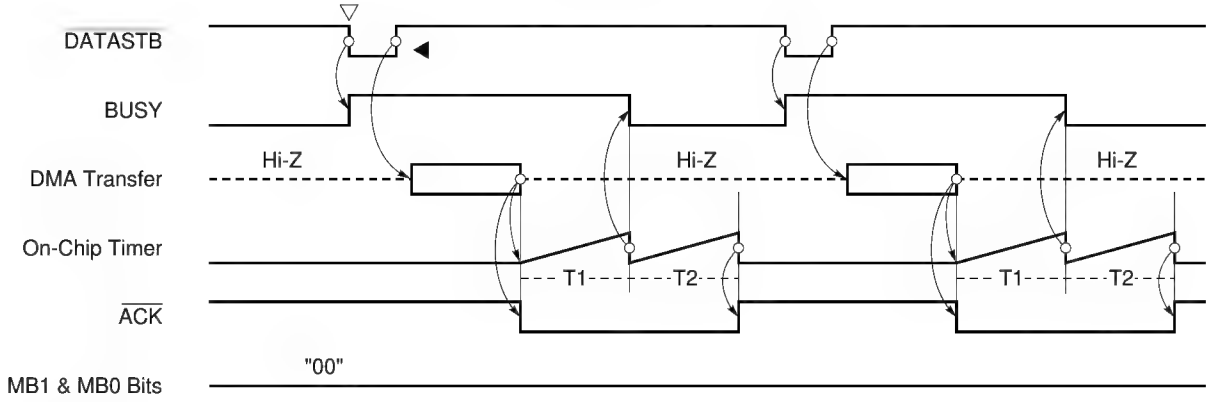


Figure 8-5. Timing Chart (STBT=1, BSYT=1)

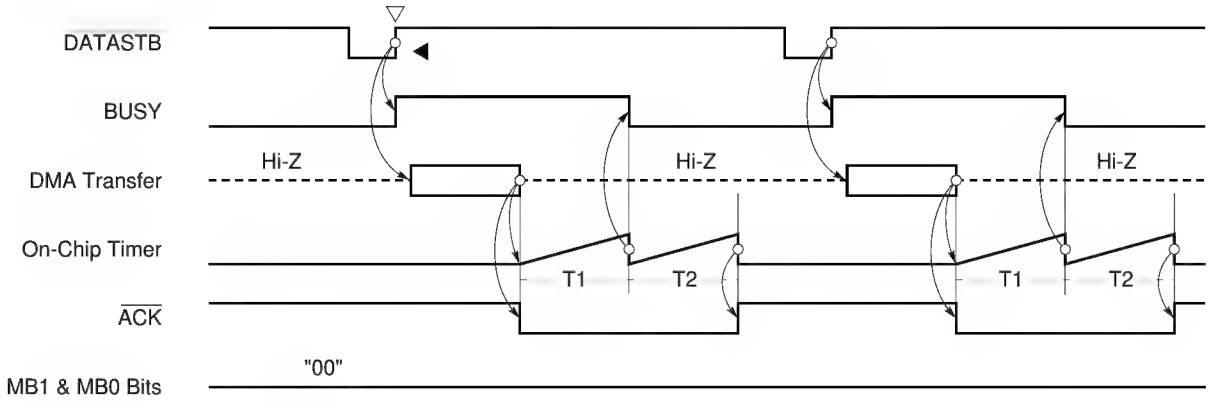


Figure 8-6. Timing Chart (STBT=0, BSYT=1)

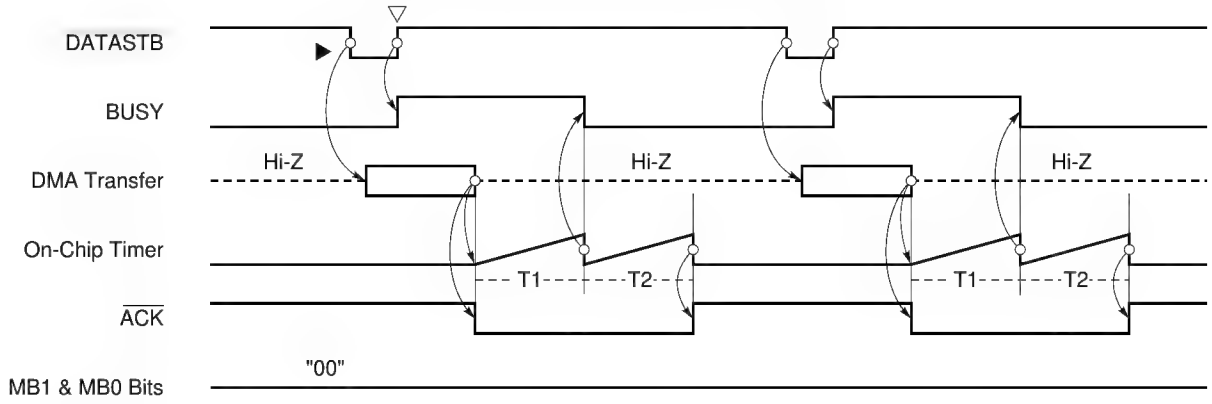


Figure 8-7. Timing Chart (ACKT1, ACKT0=00)

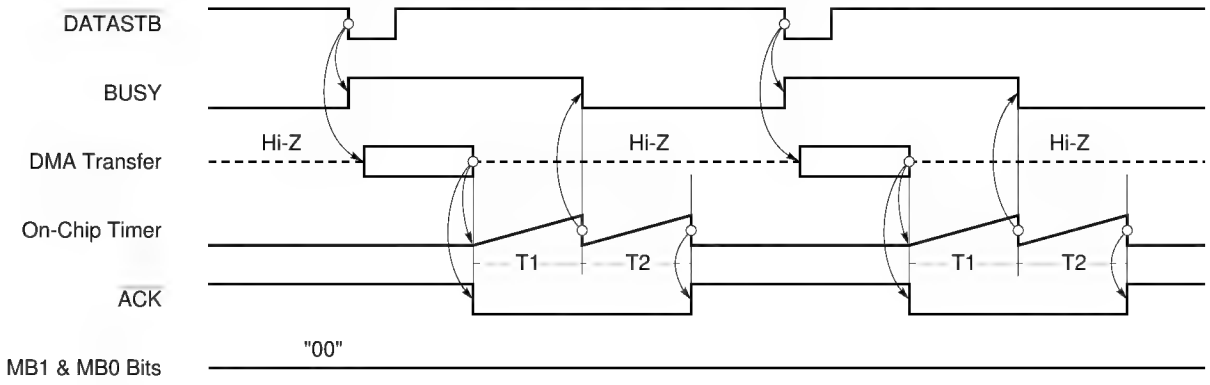


Figure 8-8. Timing Chart (ACKT1, ACKT0=01)

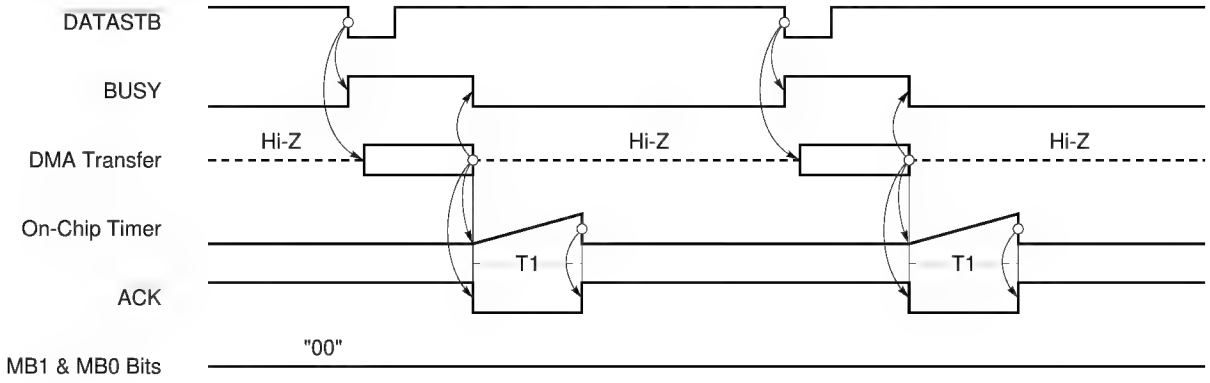


Figure 8-9. Timing Chart (ACKT1, ACKT0=10)

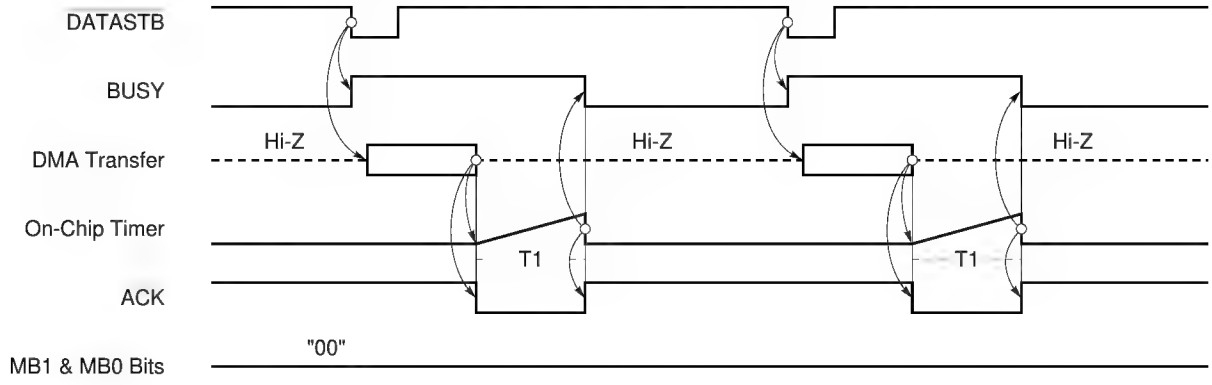


Figure 8-10. Timing Chart (ACKT1, ACKT0=00)

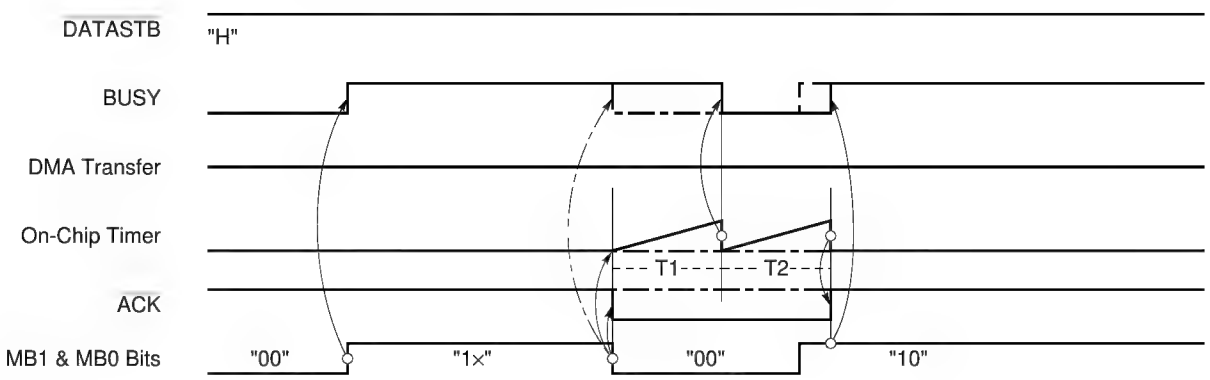


Figure 8-11. Timing Chart (ACKT1, ACKT0=01)

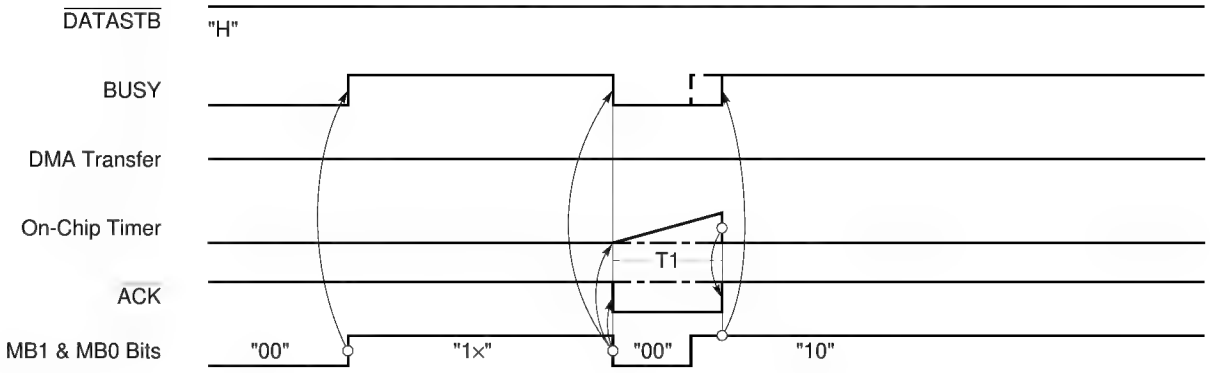
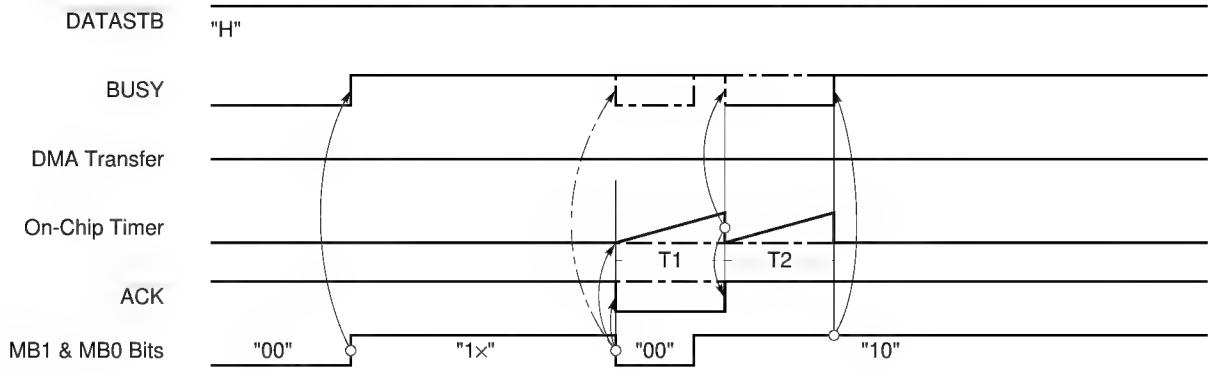


Figure 8-12. Timing Chart (ACKT1, ACKT0=10)



Remark — — — : ACKM=0
——— : ACKM=1

Figure 8-13. Timing Chart (ACKT1, ACKT0=00)

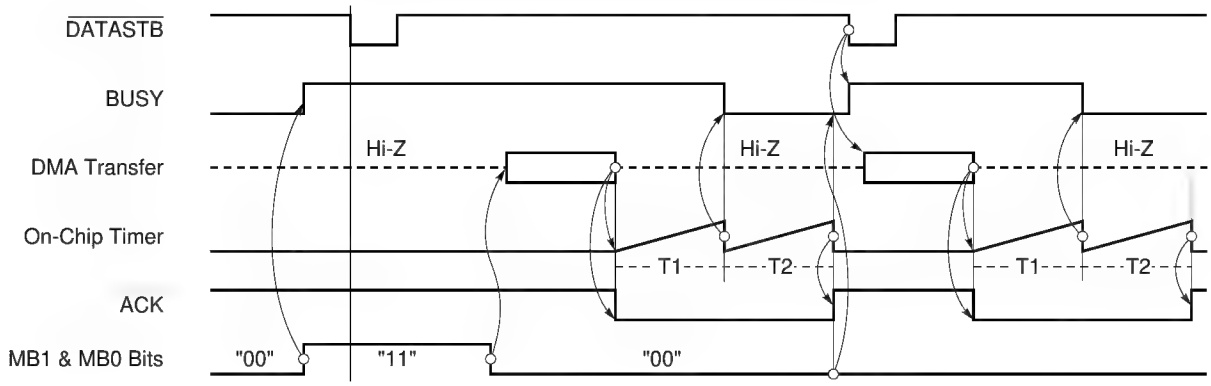


Figure 8-14. Timing Chart (ACKT1, ACKT0=01)

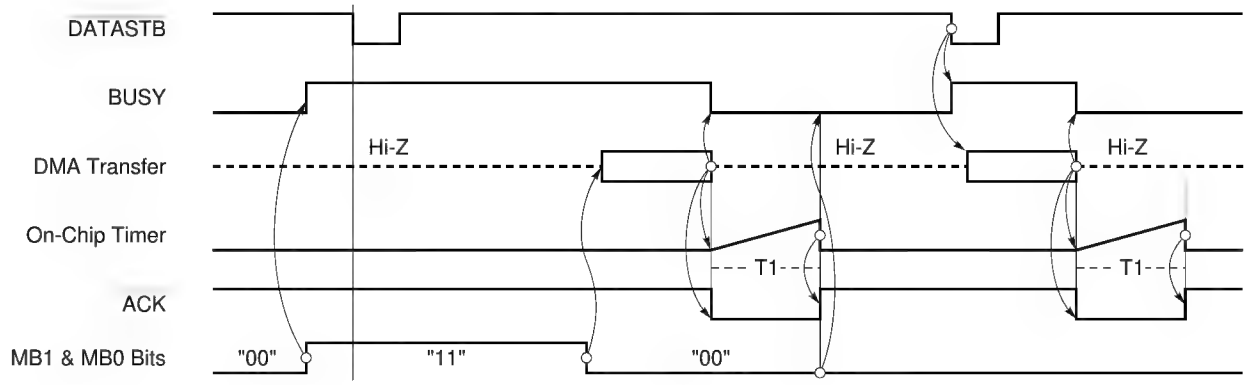


Figure 8-15. Timing Chart (ACKT1, ACKT0=10)

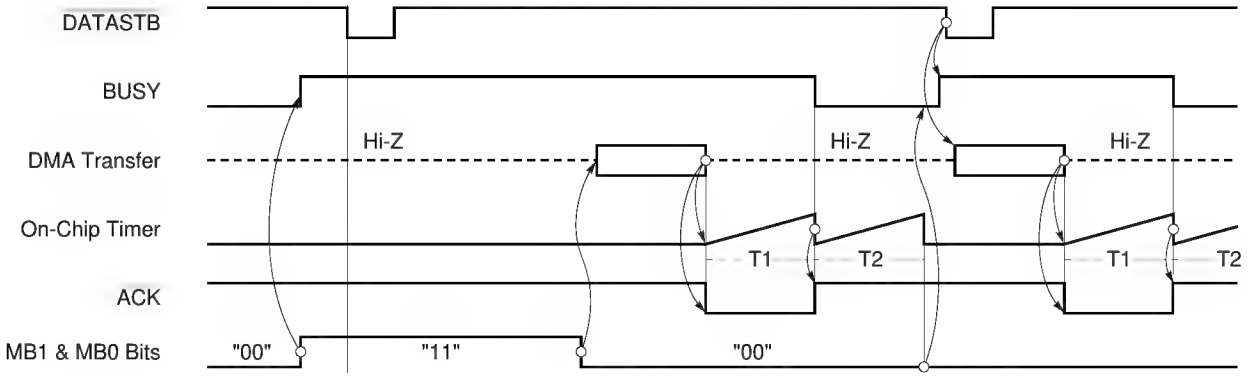


Figure 8-16. Timing Chart (ACKT1, ACKT0=00)

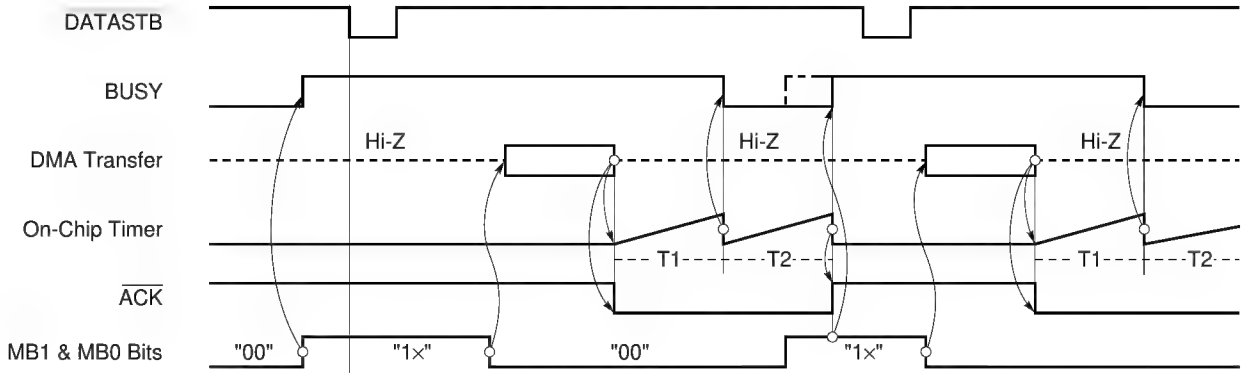


Figure 8-17. Timing Chart (ACKT1, ACKT0=01)

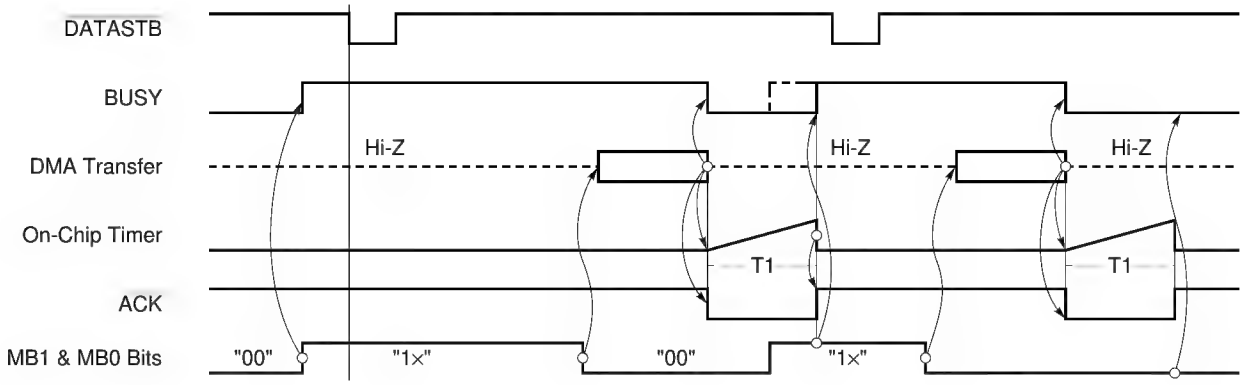
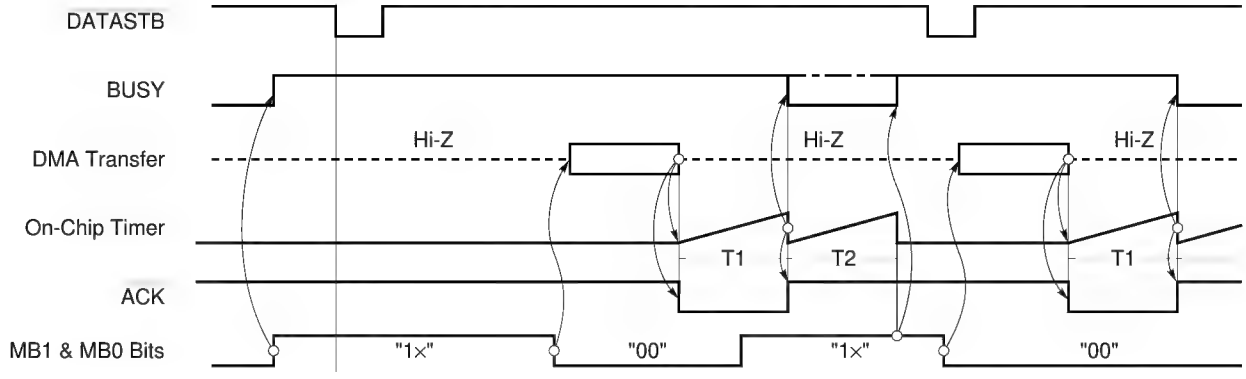


Figure 8-18. Timing Chart (ACKT1, ACKT0=10)



Remark ——— When MB1, MB0 are set to “10”
— — — When MB1, MB0 are set to “11”

Figure 8-19. Timing Chart (ACKT1, ACKT0=00)

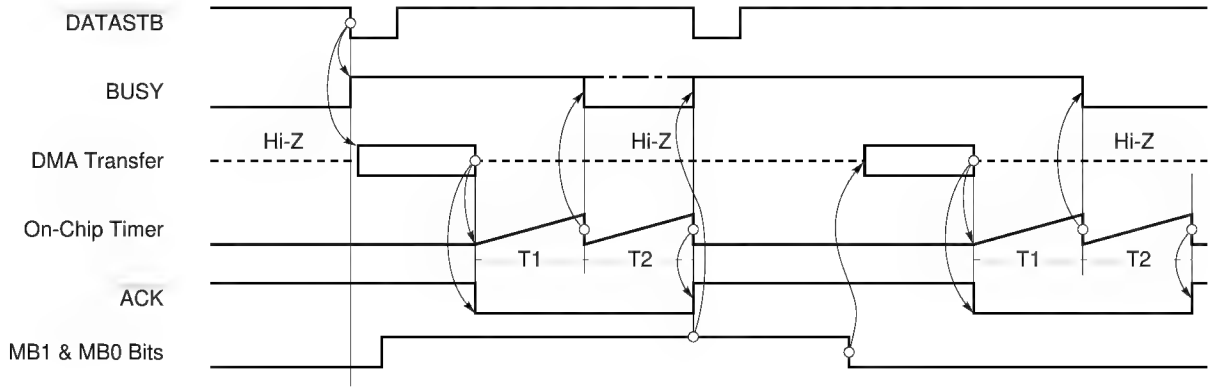


Figure 8-20. Timing Chart (ACKT1, ACKT0=01)

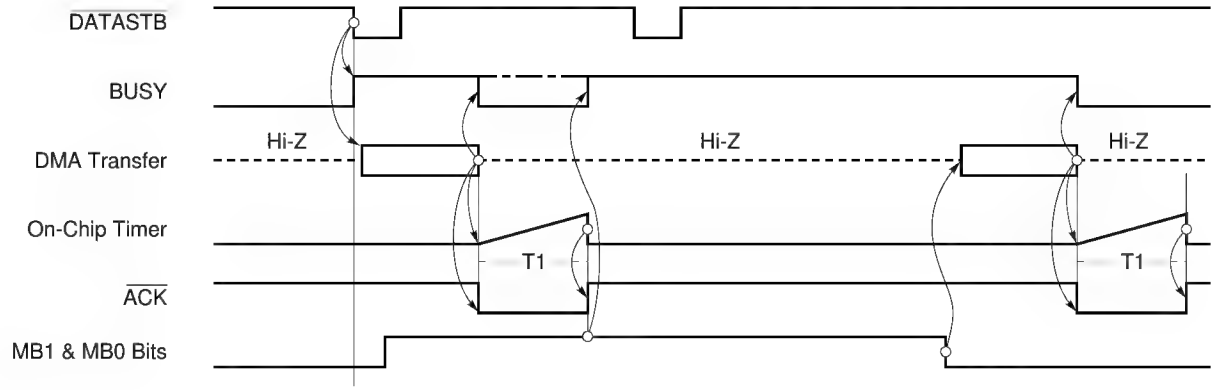
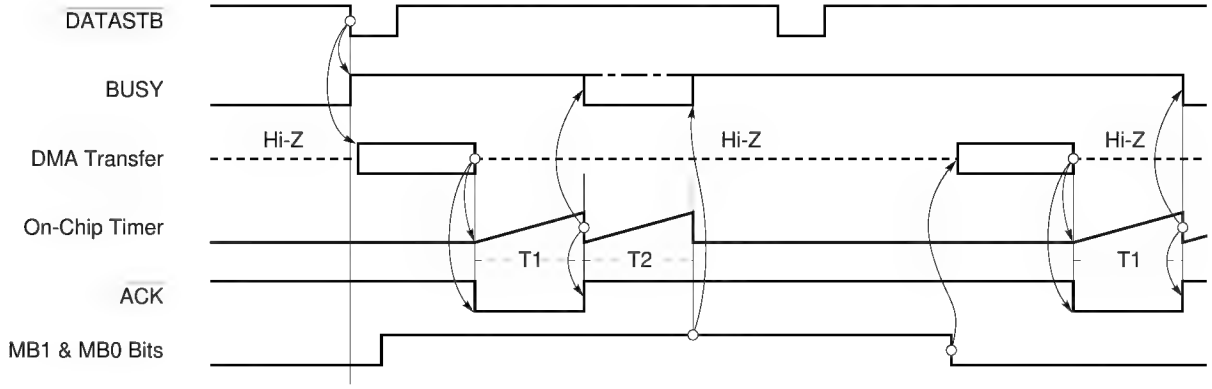


Figure 8-21. Timing Chart (ACKT1, ACKT0=10)



Remark ——— When MB1, MB0 are set to “10”
— — — When MB1, MB0 are set to “11”

Figure 8-22. Timing Chart (ACKT1, ACKT0=00)

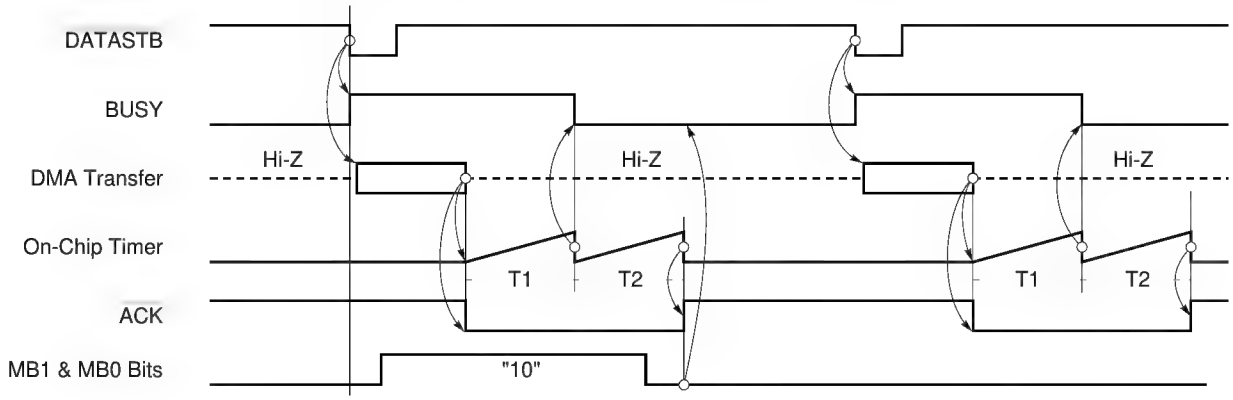


Figure 8-23. Timing Chart (ACKT1, ACKT0=01)

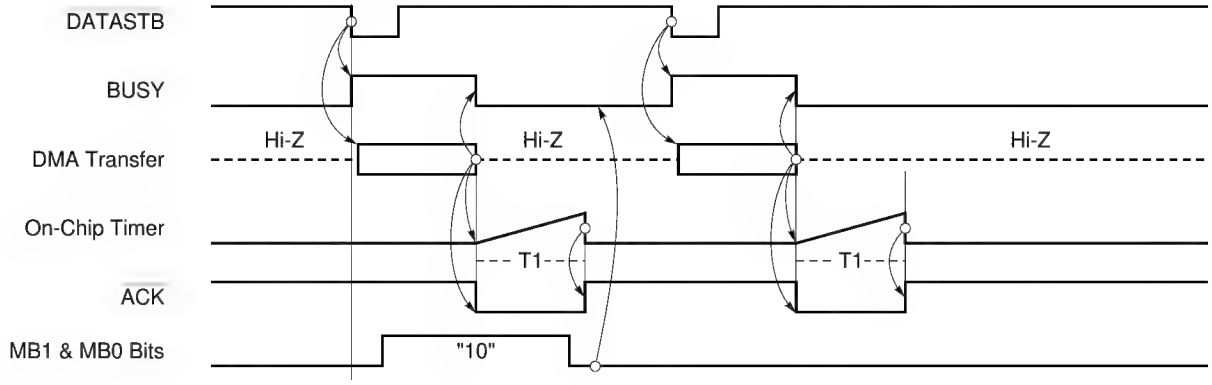


Figure 8-24. Timing Chart (ACKT1, ACKT0=10)

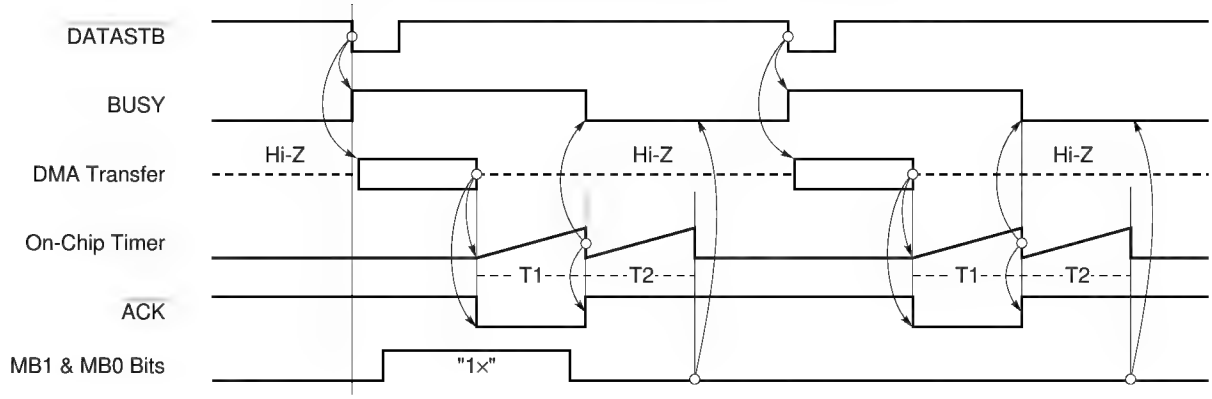


Figure 8-25. Timing Chart (ACKT1, ACKT0=00)

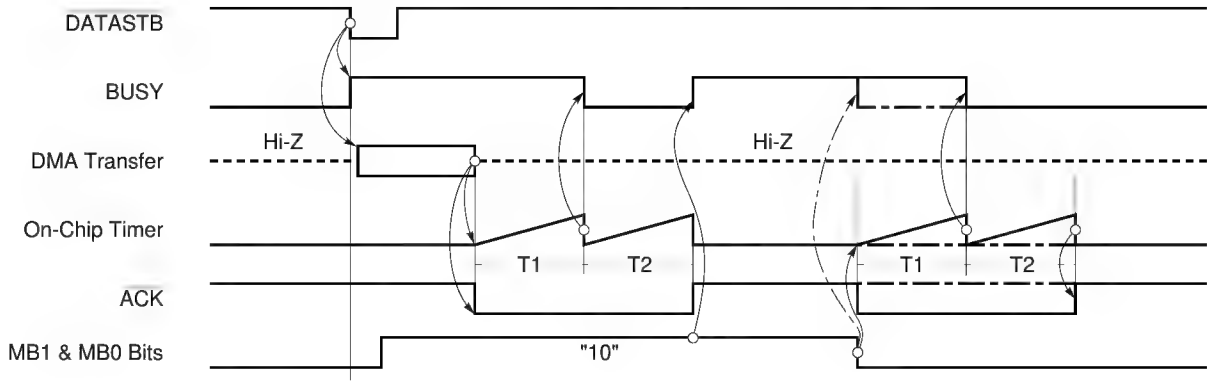


Figure 8-26. Timing Chart (ACKT1, ACKT0=01)

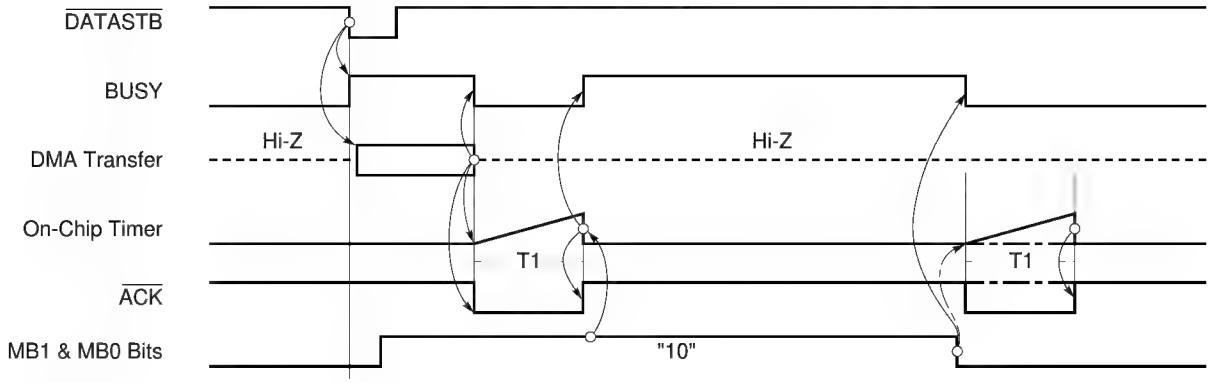
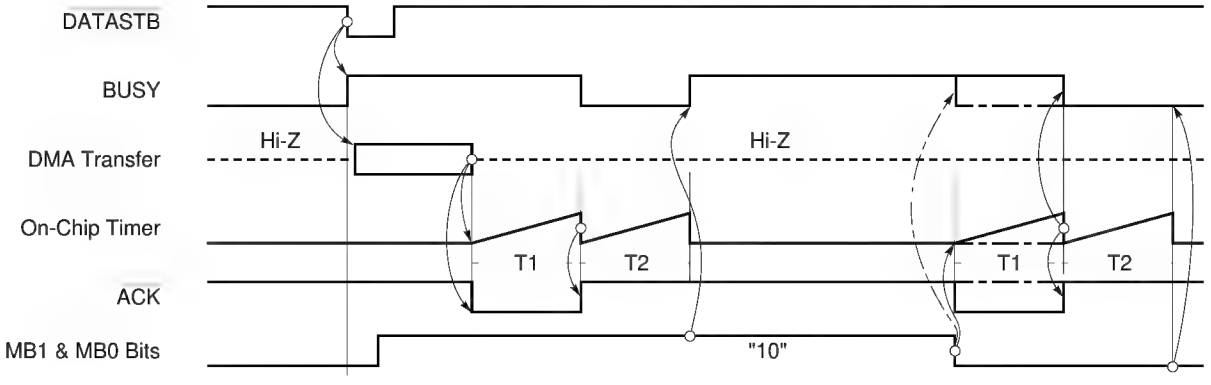


Figure 8-27. Timing Chart (ACKT1, ACKT0=10)



Remark ——— ACKM = 0
——— ACKM = 1

Figure 8-28. Timing Chart (ACKT1, ACKT0=00, ACKM=0)

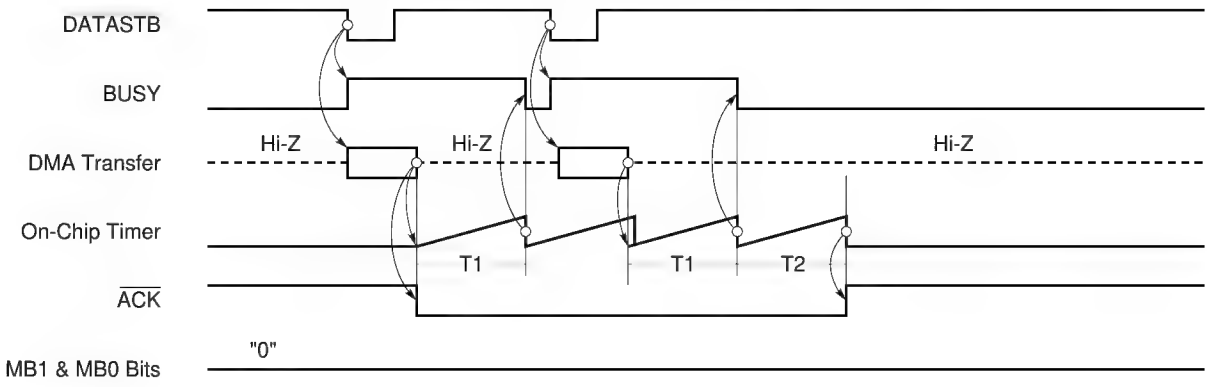


Figure 8-29. Timing Chart (ACKT1, ACKT0=01, ACKM=0)

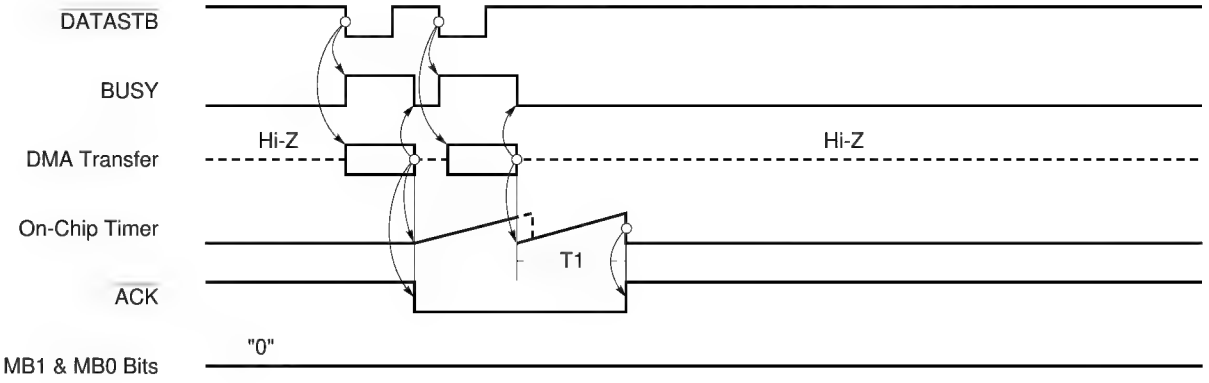


Figure 8-30. Timing Chart (ACKT1, ACKT0=10, ACKM=0)

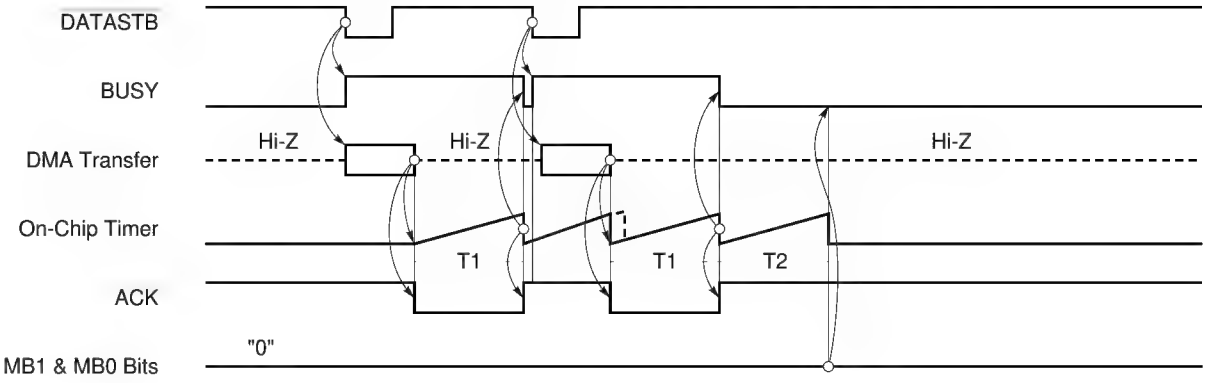


Figure 8-31. Timing Chart (ACKT1, ACKT0=00, ACKM=0)

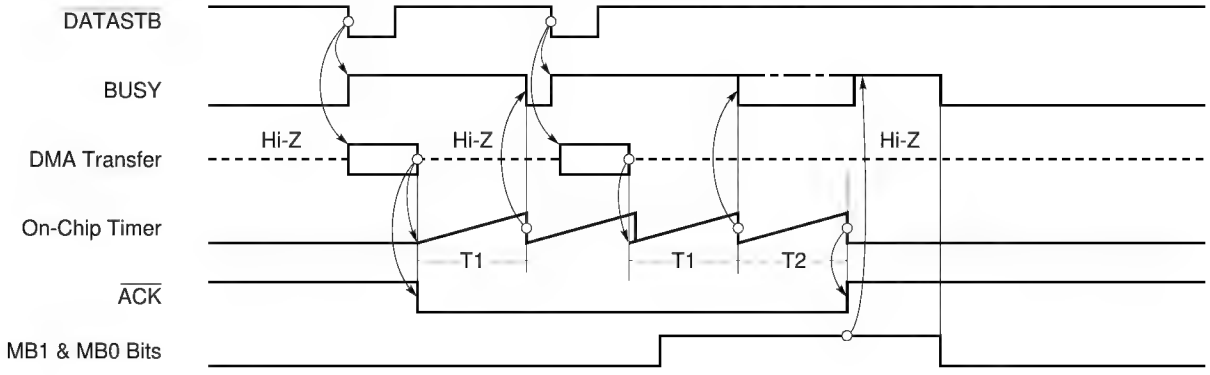


Figure 8-32. Timing Chart (ACKT1, ACKT0=01, ACKM=0)

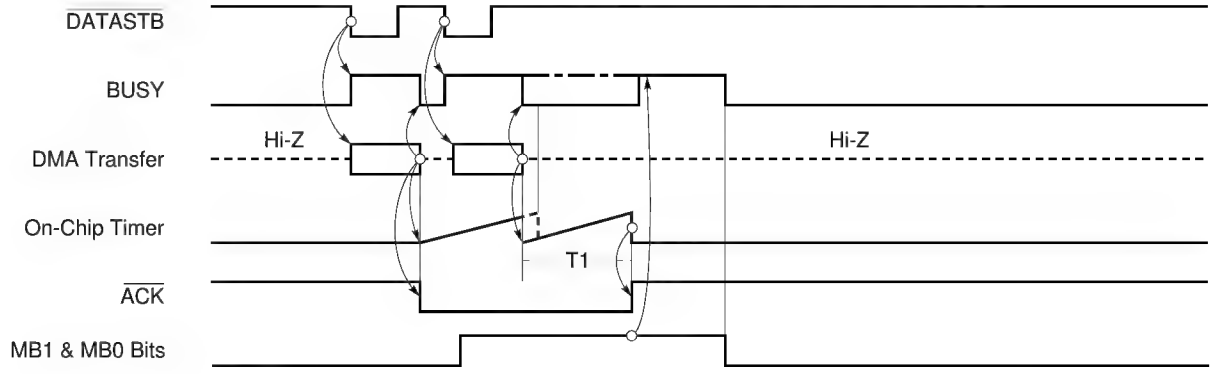
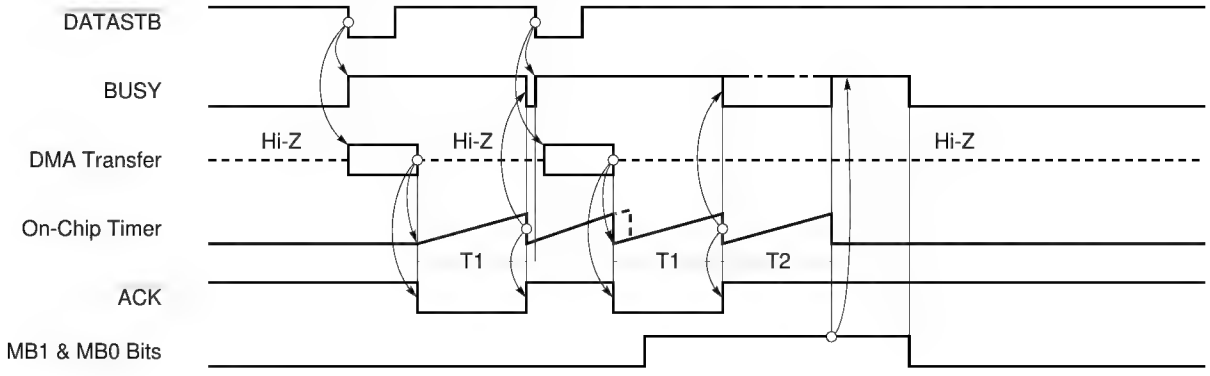


Figure 8-33. Timing Chart (ACKT1, ACKT0=10, ACKM=0)



Remark ——— When MB1, MB0 are set to "10"
— — — When MB1, MB0 are set to "11"

[MEMO]

CHAPTER 9 TIMER FUNCTIONS

The V55PI timer unit can be used as an interval timer, free running timer and event counter. P7 can so be used as a real-time output port synchronized to interrupt requests generated from the timer. This chapter describes the normal timer functions.

9.1 Features

- 16-bit timer x 4
- Two count clock sources
  - System clock divided output (ϕ/8, ϕ/32: system clock ϕ) can be selected.
  - External pulse from TI pin can be selected.
- External count output signal (TOn output)
- Three on-chip 16-bit capture registers (Triggered by external interrupt input signals INTP0 to INTP2.)
- Six timer unit dedicated interrupt sources.

9.2 Configuration

The timer unit block diagram is shown in Figure 9-1. The timer unit roughly consists of four independent 16-bit timers. Each timer can be used as shown below.

Table 9-1. Types of Timer Units

		Timer 0	Timer 1	Timer 2	Timer 3
Type	Free running timer	√	√	–	–
	Event counter	–	√	–	–
	Interval timer	–	√	√	√
Output	Number of interrupt requests	2	2	1	1
	Real-time output trigger	√	–	–	–
	Number of external output pins	2	–	2	1

(1) Timer 0

Timer 0 consists of 16-bit timer register 0 (TM0), 16-bit compare registers (CM00, CM01), and 16-bit capture registers (CT00, CT01) and functions as a 16 bit free running timer.

(2) Timer 1

Timer 1 consists of a 16-bit timer register (TM1), 16-bit compare registers (CM10, CM11), and a 16-bit capture register (CT10) and functions as a 16-bit free running timer, event counter, and interval timer.
It can be used as s free running timer by disabling the counter clear operation by CM10 coincidence signal.
The 16-bit interval timer function is enabled by counter clear operation by CM10 coincidence signal.

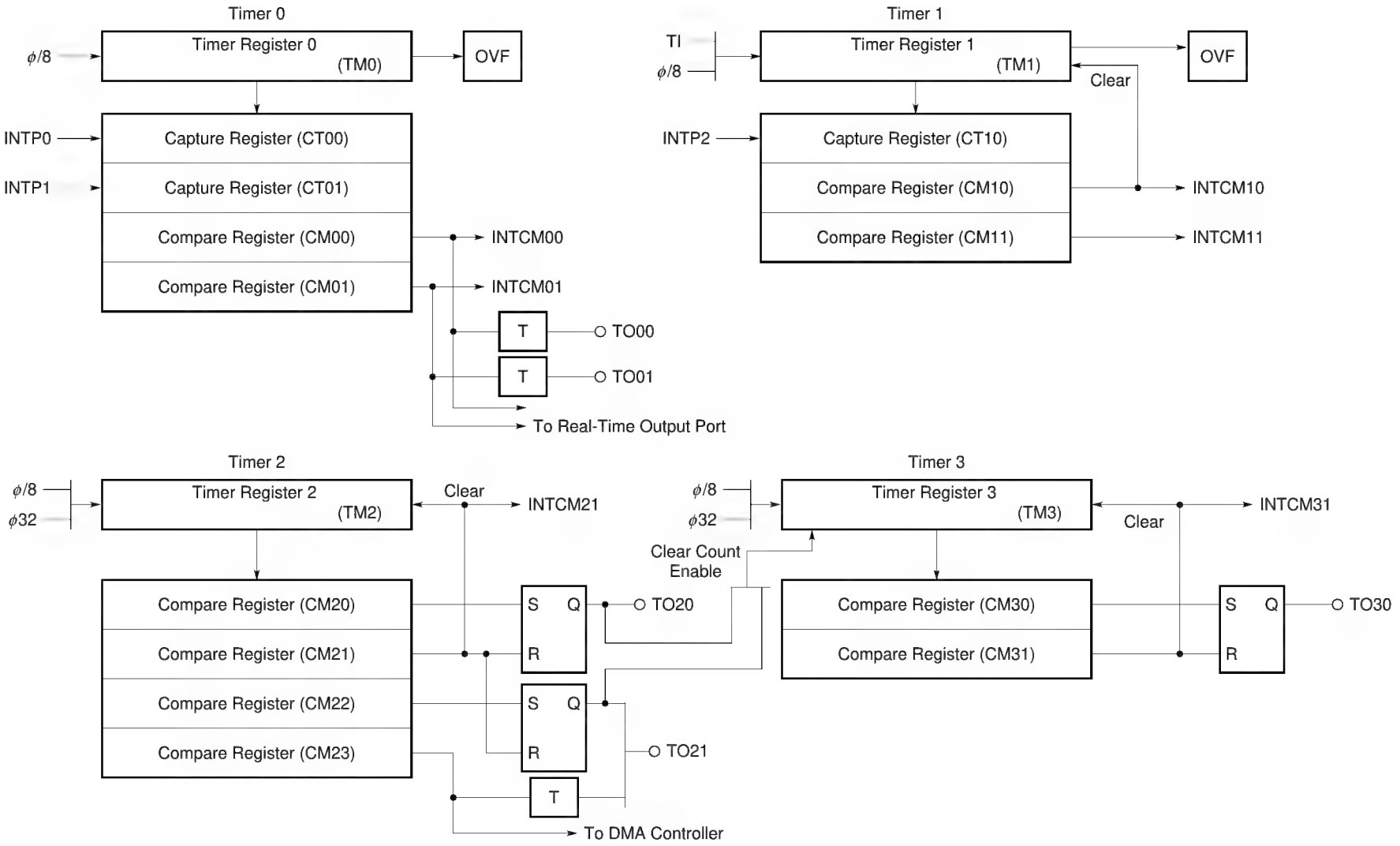
(3) Timer 2

Timer 2 consists of 16-bit timer register 2 (TM2) and 16-bit compare registers (CM20, CM21, CM22, CM23) and functions as a 16-bit interval timer.

(4) Timer 3

Timer 3 consists of 16-bit timer register 3 (TM3) and 16-bit compare registers (CM30, CM31) and functions as a 16-bit interval timer.

Figure 9-1. Timer Unit Block Diagram



9.3 Operation

The functions of each timer are shown in Table 9-2.

Table 9-2. Functions of Each Timer

			Timer 0	Timer 1	Timer 2	Timer 3
Function	Function Count		√	√	√	√
	Capture		√	√	–	–
	Compare		√	√	√	√
	Timer output	Toggle output	√	–	√	–
		Set/reset output	–	–	√	√
	Cascade connection		–	–	√	

9.3.1 Count function

There are two types of count operations. They are shown below.

- **Free-running timer (See Figure 9-2)**
  Counts the source clock with the 16-bit timer register (TM).
  The counting operation is started by setting the timer control register CE bit.
  If the CE bit is set (1) again when it has already been set, the TM register is not cleared and the count operations continued.
  When counting reaches FFFFH, borrow from the TM register occurs and the OVF flag of the TMC register is set. When borrow occurs, the TM register is cleared and count operations are continued.

- **Interval timer (See Figure 9-3)**
  Interrupt requests are repeatedly generated with the count time preset as the interval.
  Counts the source clock with the 16-bit timer register (TM).
  The counting operation is started by setting the timer control register CE bit.
  When the value counted up by the TM register and the CM register value coincide, the TM register is cleared and count operations are continued.

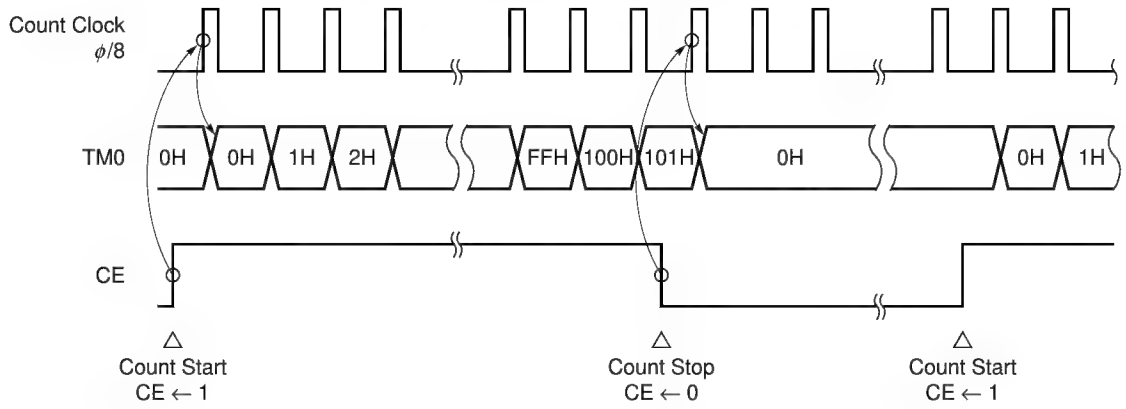
The count clock and count time for each system clock frequency are shown in Table 9-3.

Table 9-3. Timer Count Time

Count clock	System clock (ϕ)				Remarks
	8 MHz	10 MHz	12.5 MHz	16 MHz	
ϕ /8	1 μs - 65.4 ms	0.8 μs - 52.4 ms	0.6 μs - 42.0 ms	0.5 μs - 32.7 ms	Count time = [count clock/(compare register value+1)]$^{-1}$
ϕ /32	4 μs - 262 ms	3.2 μs - 209.6 ms	2.6 μs - 167.8 ms	2 μs - 131 ms	

Figure 9-2. Free Running Timer Operation (Timer 0) (1/2)

(a) When count start → count stop → count start



(b) When CE bit "1" is written again after count start

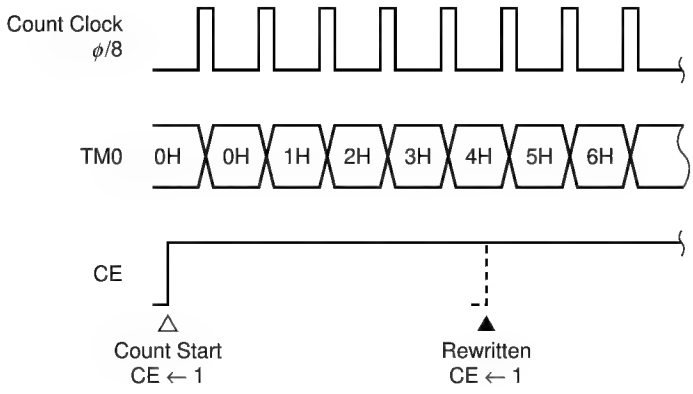


Figure 9-2. Free Running Timer Operation Timer 0 (2/2)

(c) When TMO is FFFFH

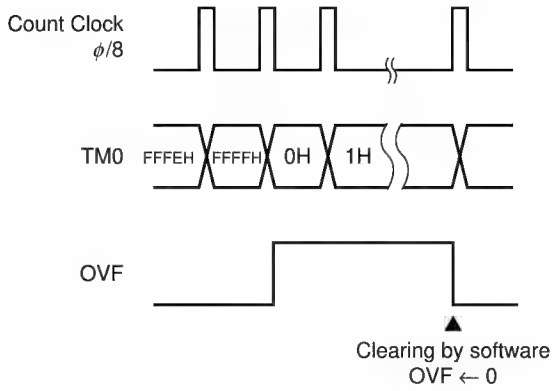
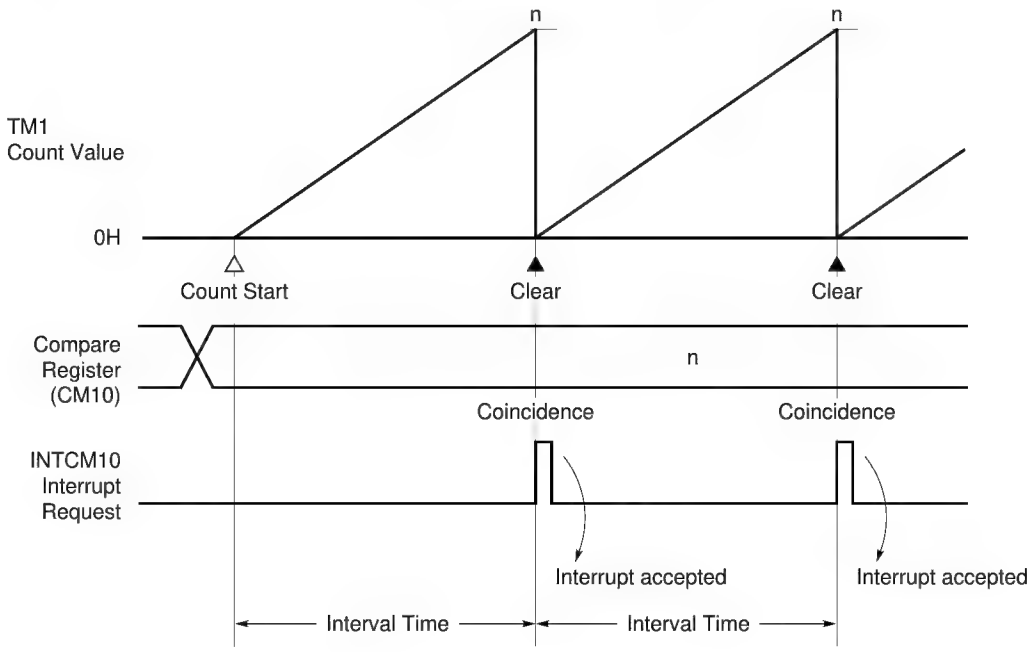


Figure 9-3. Interval Timer Operation (When CM10 is used as the compare register)



Remark Interval Time = (n+1) × x/ϕ

$0 \leq n \leq$ FFFFH

x = 8, 32

(1) Timer 0 operation

Operates only as the free running timer

The internal clock (ϕ/8) is used as the source clock.

The register used:

- Timer register: TM0

(2) Timer 1 operation

Operates as the free running timer and interval timer.

- Free-running timer: TMC0 register CLR bit=1
- Interval timer: TMC0 register CLR bit=0

The interval clock (ϕ/8) or external clock (TI pin input) can be selected as the source clock.

The registers used are:

- Timer register: TM1
- Compare register: CM10

(3) Timer 2 operation

Operates only as the interval timer.

The internal clock ϕ/8 or ϕ/32 can be selected as the source clock.

The registers used are:

- Timer register: TM2
- Compare register: CM21

(4) Timer 3 operation

The timer 3 can be connected in cascade form to timer 2.

Operates only as the interval timer.

The internal clock ϕ/8 or ϕ/32 can be selected as the source clock.

The registers used are:

- Timer register: TM3
- Compare register: CM31

If connected to timer 2 using the cascade mode (TMC1 register CCLR bit=1), the start and stop of count operations can be controlled using the set/reset output of timer 2 (See **9.3.5 Cascade Connection**).

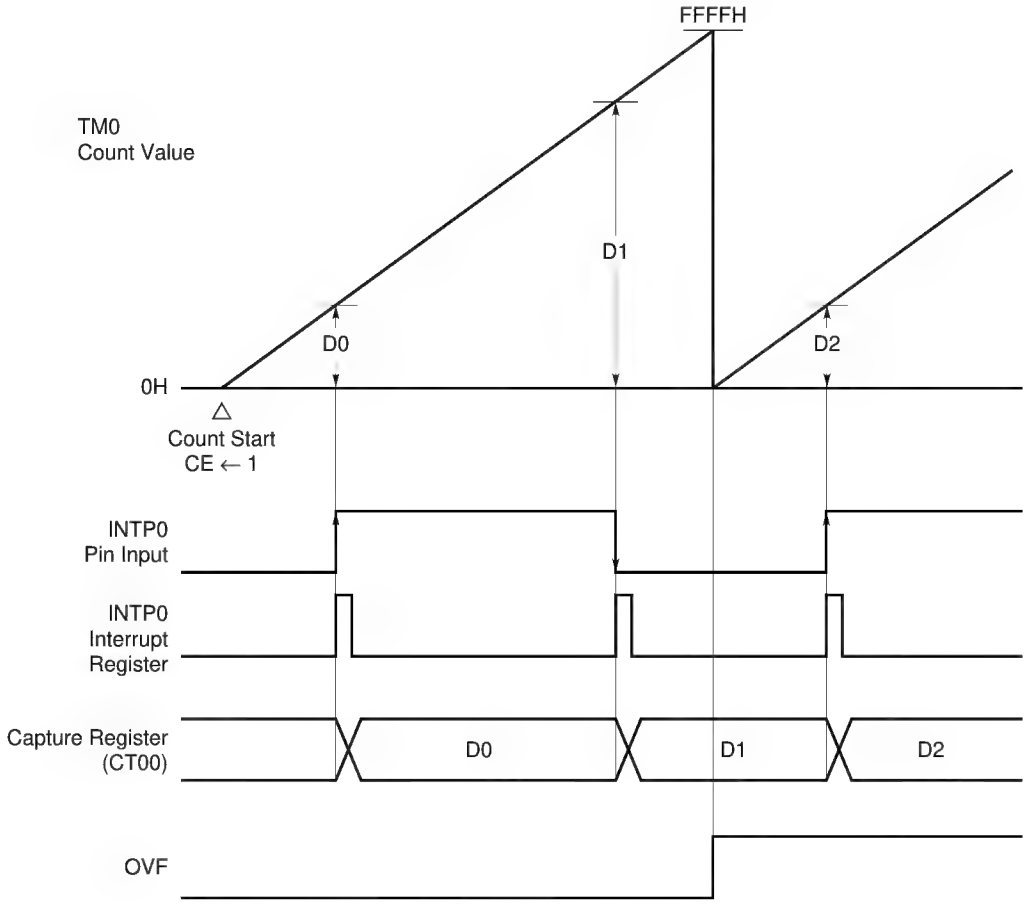
9.3.2 Capture function

The pulse width and input pulse period of the input signal can be obtained with this function.

When the capture trigger signal (external interrupt dual function pin INTP) is generated, the TM register value at that time is latched with the capture register (CT) (capture operations). The latch value is kept in the capture register (CT) until the next capture operation is executed.

The effective edge of the capture trigger is set by the external interrupt mode register 1 (INTM1) (See **5.6 External Interrupt**). If both the rising edge and falling edge are set as capture triggers, the width of pulses input externally is measured. If the capture trigger is generated with either edge, the period of the input pulse is measured.

Figure 9-4. Capture Operation (Timer 0)



Remark Dn: Count value of TM0 (n=0, 1, 2, ...)

(1) Timer 0 operation

Two capture registers are equipped.

Capture Register	Trigger Signal	Timer Register
CT00	INTP0	TM0
CT01	INTP1	

(2) Timer 1 operation

One capture register is equipped.

Capture Register	Trigger Signal	Timer Register
CT10	INTP2	TM1

9.3.3 Compare function

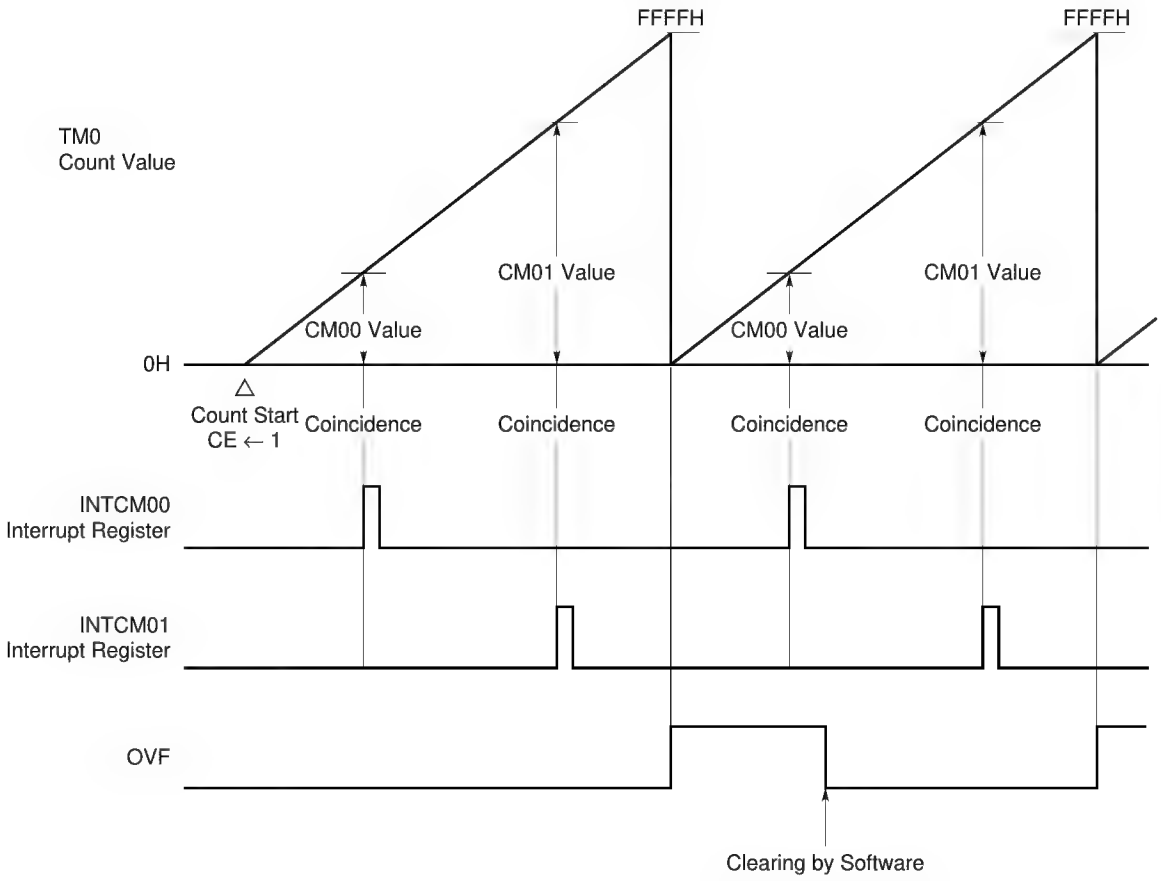
Compares the value of the preset compare register (CM) with the count value of the timer register (TM). The following three types of operations are performed when CM and TM values coincide.

- Interrupt request (INTCM) is generated.
- Timer output request is generated.
- After the TM register is cleared, count operations are continued.

The TM register is cleared only when the next compare registers are used.

Compare Register	Cleared TM Register
CM10	TM1
CM21	TM2
CM31	TM3

Figure 9-5. Compare Operation (Timer 0)



(1) Timer 0 operation

An interrupt request function is provided.

Compare Register	Interrupt Request	Timer Register
CM00	INTCM00	TM0
CM01	INTCM01	

(2) Timer 1 operation

An interrupt request function is provided.

Compare Register	Interrupt Request	Timer Register
CM10	INTCM10	TM1
CM11	INTCM11	

(3) Timer 2 operation

The functions of interrupt request and timer output request are provided.

Compare Register	Interrupt Request	Timer Output	Timer Register
CM20	None	TO20 (Set)	TM2
CM21	INTCM21	TO21/TO21 (Reset)	
CM22	None	TO21 (Set)	
CM23	None	TO21 (Toggle)	

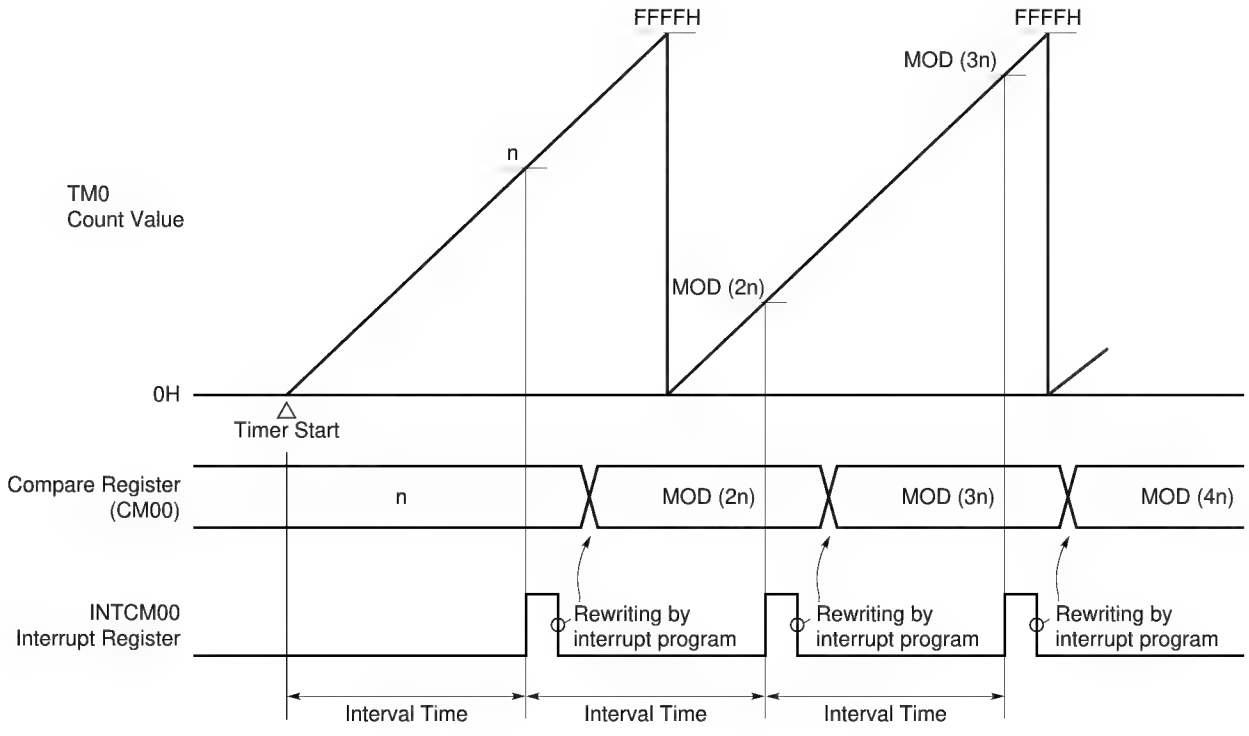
Timer outputs TO20 and TO21 can be connected to timer 3 in cascade form.

(4) Timer 3 operation

The functions of interrupt request and timer output request are provided.

Compare Register	Interrupt Request	Timer Output	Timer Register
CM30	None	TO30 (Set)	TM3
CM31	INTCM31	TO30 (Reset)	

Figure 9-6. When Timer 0 is Used as Interval Timer



Remark Interval Time=n × 8/ϕ, 1≤n≤FFFFH

9.3.4 Timer output function

By setting the corresponding bit to the control mode using the port 2 mode control register (PMC2), this pin can be used as the timer unit output pin.

The timer output function has a toggle output and set/reset output. Specify which function to use with the timer output control register 1 (TOC0) MOD bit (See **9.4.2 Timer output control registers (TOC0 and TOC1)**).

- **Toggle output**

  The toggle output synchronizes with the timer output generated when the compare register (CM) and timer register (TM) coincide and inverts the level of the timer output pin (TO).

- **Set/reset output**

  The TO pins (TO20, TO21, TO30) are set (1) by the signal generated when the values of the compare registers (CM20, CM22, CM30) and timer registers (TM2, TM3) coincide. These TO pins (TO20, TO21, TO30) are reset (0) by the signal generated when the values of the compare registers (CM21, CM31) and timer registers (TM2, TM3) coincide.

(1) Timer 0 operation

A toggle output function is provided.

This coincidence detection is used as the toggle of the real-time output port.

Compare Register	Timer Output Terminal	Timer Register
CM00	TO00	TM0
CM01	TO01	

(2) Timer 2 operation

A toggle output and set/reset output function is provided.

If used as the set/reset function, the TM2 register is cleared if it has been detected as coinciding CM21, and as a result, two-phase pulse is output from the TO20 and TO21 pins.

Compare Register	Timer Output Terminal	Timer Register
CM23	TO21 (Toggle)	TM2
CM20	TO20 (Set)	
CM21	TO20/TO21 (Reset)	
CM22	TO21 (Set)	

(3) Timer 3 operation

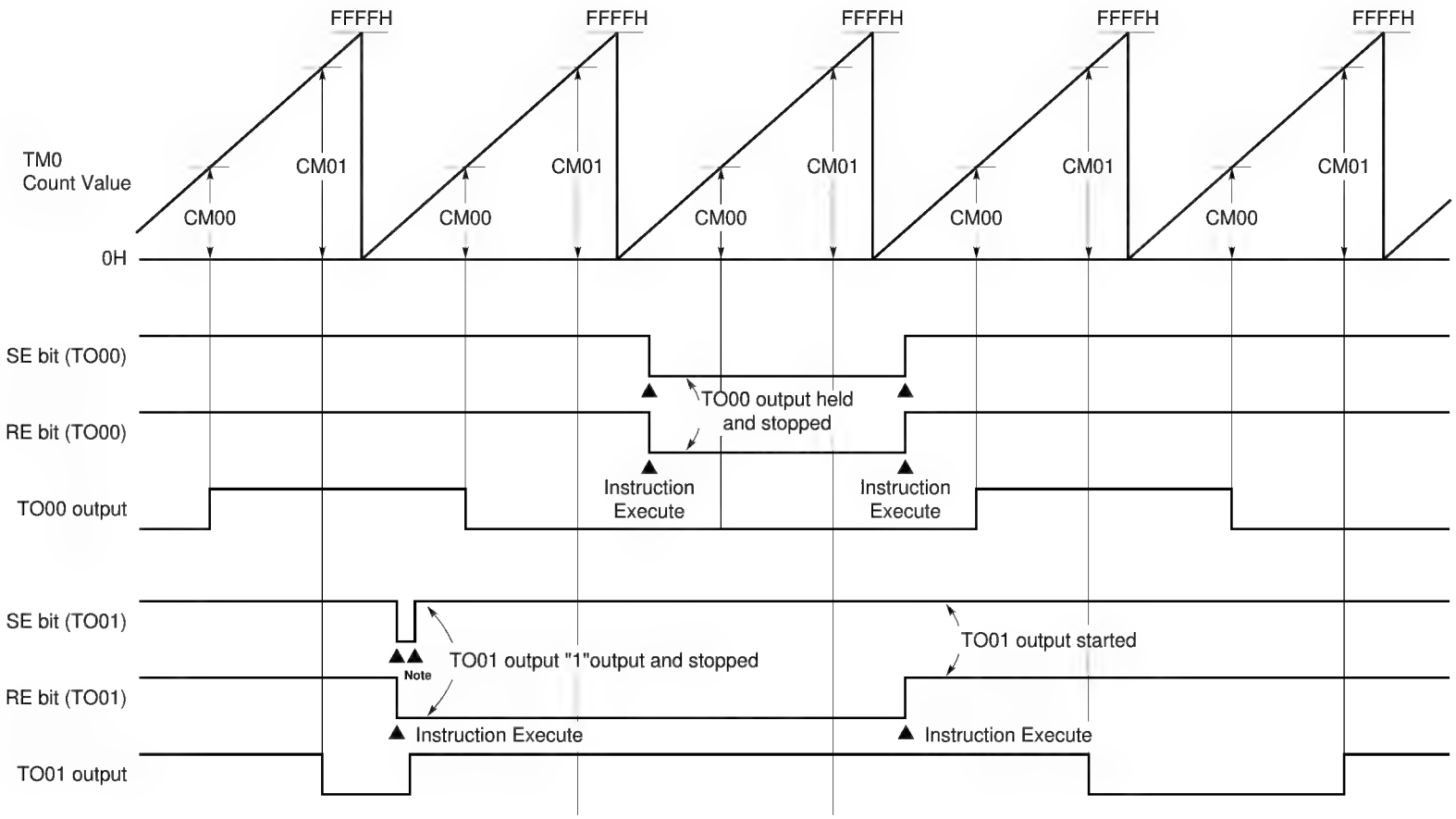
A set/reset output function is provided.

This timer 3 operation is performed only when the cascade connection mode is not set. See **9.3.5** for details of the cascade connection mode.

Compare Register	Timer Output Terminal	Timer Register
CM30	TO30 (Set)	TM3
CM31	TO30 (Reset)	

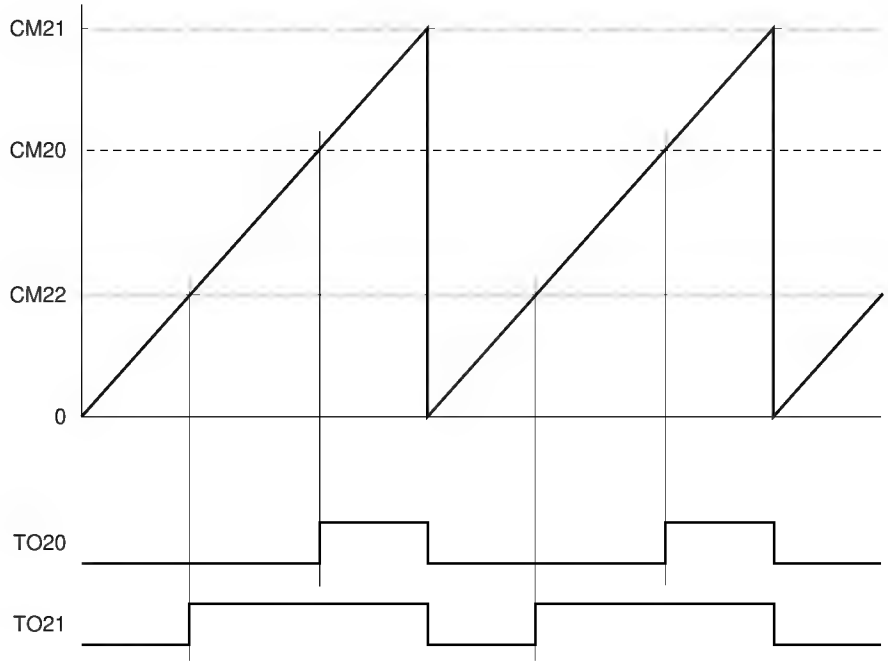
Figure 9-7 shows the toggle output operations and Figures 9-8 shows the set/reset output operations.
The timer output pin is controlled by the combination of the SE bit and RE bit of the timer output control register.

Figure 9-7. Toggle Output by Timer 0



Note Change of the TO output signal level, change of the timer output operations, and permit control cannot be executed together. Therefore, timer operations stop and change of the TO output level are performed separately in two stages.

Figure 9-8. Set/Reset Output by Timer 2



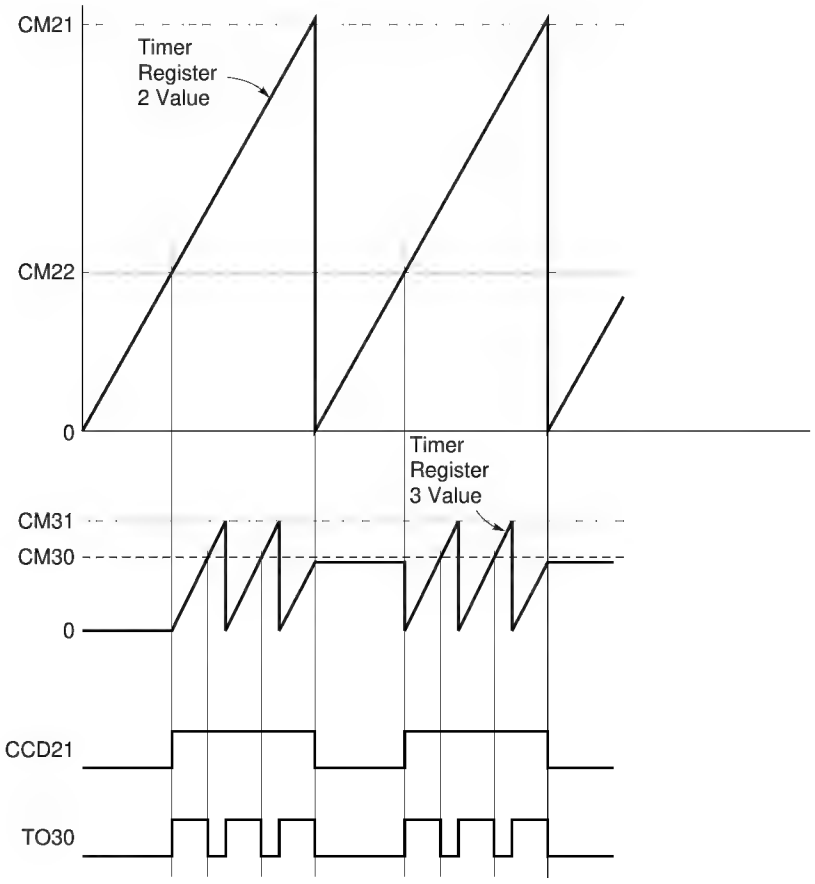
9.3.5 Cascade connection

Timer 2 and timer 3 can be connected in cascade by setting (1) the timer control register (TMC1) CCLR bit.

While the timer 2 set/reset output CCD20 (corresponds to the TO20 pin) is low, the timer 3 counting operation stops and the output from the TO30 pin is masked. The timer 3 counting operation is started by the rising edge of the CCD20 output and timer 3 set/reset is output from the TO30 pin while CCD20 is high. Timer 3 is cleared by the rising edge of CCD20. Set/reset output CCD21 (corresponds to the TO21 pin) from timer 2 can be used similarly.

The TO30 output waveform when cascaded by CCD21 signal is shown in Figure 9-9.

Figure 9-9. TO30 Output Waveform by Cascade Connection



Cautions 1. In the cascading mode, the TO pin output level is the reverse of the normal level. It is set when the CM30 register is detected to coincide (0) and reset when the CM31 register is detected to coincide (1).

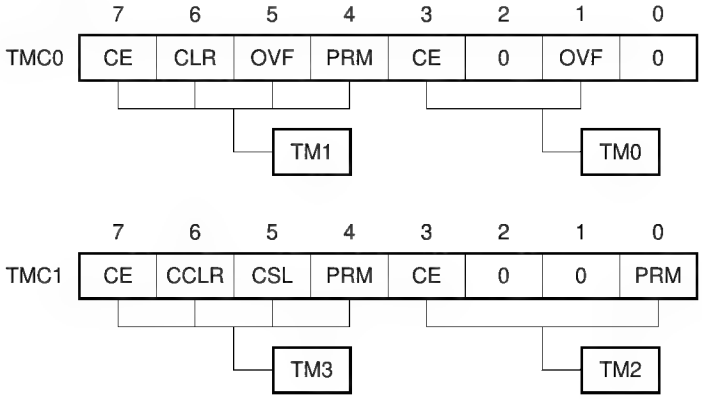
2. In the cascading mode, if the timer 2 CE bit is changed from 1 to 0, timer 3 also stops counting. This is because timer 3 stops counting while CCD20 or CCD21 output is low.

9.4 Control Registers

9.4.1 Timer control registers (TMC0, TMC1)

The TMC0 register is an 8-bit register that specifies the timer 0 and timer 1 operation mode and selects the timer 0 and timer 1 count clock. Similarly, the TMC1 register specifies the timer 2 and timer 3 operation and selects the timer 2 and timer 3 count clock. Timers 0 to 3 can be started and stopped simultaneously by word accessing the TMC0 and TMC1 registers.

After reset, the value of the TMC0 and TMC1 registers becomes 00H. Hereafter shaded area in a table indicates status after a reset.



PRM **TMn count clock source selection (ϕ: system clock)**

Timer	PRM	Count Clock
TM0	0	ϕ/8 (fixed)
TM1	0	ϕ/8
	1	External clock (TI)
TM2	0	ϕ/8
	1	ϕ/32
TM3	0	ϕ/8
	1	ϕ/32

OVF **Timer overflow status flag**

Timer	OVF	Status
TM0	0	Carry signal not generated from TM0
	1	Carry signal generated from TM0
TM1	0	Carry signal not generated from TM1
	1	Carry signal generated from TM1

CLR **Timer 1 (TM1) operation mode specification**

CLR	Mode	Operation
0	Interval timer	Clears the counter with the CM10 register coincidence signal.
1	Free running timer	Continues counting without regard to the CM10 register coincidence signal.

CE **Timer n operation control flag**[Note]

CE	Operation
0	Clear and stop TMn
1	Start TMn counting operation

Note When CE=0, writing to timer/counter by instruction is enabled.
When CE=1, the value is not written.

CCLR **CSL** **TO30 pin output operation specification**

CCLR	CSL	Timer 3 set/reset output control
0	0	Set/reset output without CCD20, CCD21 output control
0	1	Setting prohibited
1	0	Select CCD20 as TO30 pin set/reset output control signal
1	1	Select CCD21 as TO30 pin set/reset output control signal

Operation in CCLR = 1 mode

While CCD20 (or CCD21) is low, timer 3 stops counting. During this period, the TO30 pin is also masked and outputs low level.

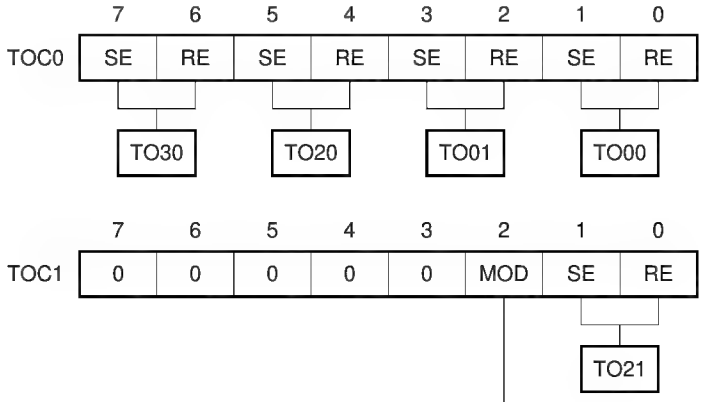
The rising edge of CCD20 (or CCD21) starts timer 3. While CCD20 (CCD21) is high, the set/reset signal is output from TO30 by CM31 and CM30 coincidence signal.

The falling edge of CCD20 (or CCD21) clears timer 3. (Timer 3 is cleared by clearing of the timer 3 CE bit in the CCLR = 1 mode also.)

9.4.2 Timer output control registers (TOC0, TOC1)

The TOC0 and TOC1 registers control the output level of external output pins TO00-TO01, TO20-TO21, and TO30.

After reset, the value of the TOC0 and TOC1 registers becomes 00H. After reset, the output level at each pin is made 1 by setting the port 2 mode control register (PMC2) to the control mode.



TO21 Output Signal Output Mode Select Flag

MOD	Mode	Operation
0	Toggle output mode	The CM23 register coincidence signal inverts the TO21 output level
1	Set/reset output mode	The CM22 register coincidence signal sets (1) the TO21 output level and the CM21 register coincidence signal resets (0) the TO21 output level.

TO00, TO01, TL20, TO21, TO30 Pin Control Flag

SE	RE	Output state
0	0	Hold, stop
0	1	0 output, stop
1	0	1 output, stop
1	1	Start

Caution Output pins TO00, TO01, TO20, TO21 and TO30 signal level modification and timer output operation modification and enable control cannot be executed simultaneously.

Remark The shaded area indicates status after a reset.

9.5 Interrupt Requests

9.5.1 Interrupt requests from timer unit

The timer unit generates an interrupt request and sends it to the interrupt control unit when coincidence of the contents of each timer and compare register is detected.

The timer unit generates the following six interrupts.

- INTCM00 (CM00 coincidence detection interrupt)
- INTCM01 (CM01 coincidence detection interrupt)
- INTCM10 (CM10 coincidence detection interrupt)
- INTCM11 (CM11 coincidence detection interrupt)
- INTCM21 (CM12 coincidence detection interrupt)
- INTCM31 (CM31 coincidence detection interrupt)

9.5.2 Capture trigger dual function external interrupt

The following external interrupt signals are the timer 0 and timer 1 capture trigger.

For a description of external interrupt control and the setting control registers, see **5.6 External interrupts**.

- INTP0 (capture register CT00 capture trigger)
- INTP1 (capture register CT01 capture trigger)
- INTP2 (capture register CT10 capture trigger)

9.6 Caution

- **Count Timing of TI input event**

The sampling of the event count input pin (TI) is performed every ϕ/8. The count precision is as follows.

Dividing ratio of system clock ϕ	Sampling time	
	fxx = 25 MHz	fxx = 32 MHz
ϕ = fxx/2	0.64 μs	0.50 μs
ϕ = fxx/4	1.28 μs	1.00 μs
ϕ = fxx/8	2.56 μs	2.00 μs
ϕ = fxx/16	5.12 μs	4.00 μs

fxx: Crystal oscillation frequency

<1> For an application circuit which inputs a serial clock to external event input pin TI and calculates the communi-cation speed, if the communication speed is made 64 kbps, a minimum accuracy of 15.6 μs is necessary.

<2> When sampling accuracy is taken into account, in the ϕ = fxx/2 mode, the maximum input clock frequency is 156 kHz.
However, from the standpoint of the circuit, if less than TI<ϕ/8, erroneous calculation will not occur. (This is not very suitable for application circuits which require high-speed, real-time response to external event

[MEMO]

CHAPTER 10 REAL-TIME OUTPUT PORT FUNCTION

V55PI port 7 has an on-chip real-time output port function. The contents of the port 7 buffers (P7H, P7L) can be output from timer 0 in bit units at a programmable interval.

10.1 Features

- 8-bit × 1ch or 4-bit × 2ch
- The macro service can be used for the output control timer, output buffer
- On-chip output delay function

10.2 Configuration

The real-time output port block diagram is shown in **Figure 10-1**. The real-time output port consists of the following buffer registers, output port, and control registers.

(1) Port 7 buffers (P7H, P7L)

The port 7 buffers are registers which hold the data to be output next when port 7 was set to the real-time output port mode. The contents are transferred to the real-time output port (RTP) and are output from the pin by the timing shown in **Table 10-3**.

Reset input has no affect on the contents of the port 7 buffers.

(2) Real-time output port (RTP)

The real-time output port output data is held at this output port from the port 7 buffers and is output from the pin.

This port can be read/written by 8-bit or 1-bit instruction operation. (Different from the port 7 output port.)

(3) Real-time output port delay specification register (RTPD) and delay counter

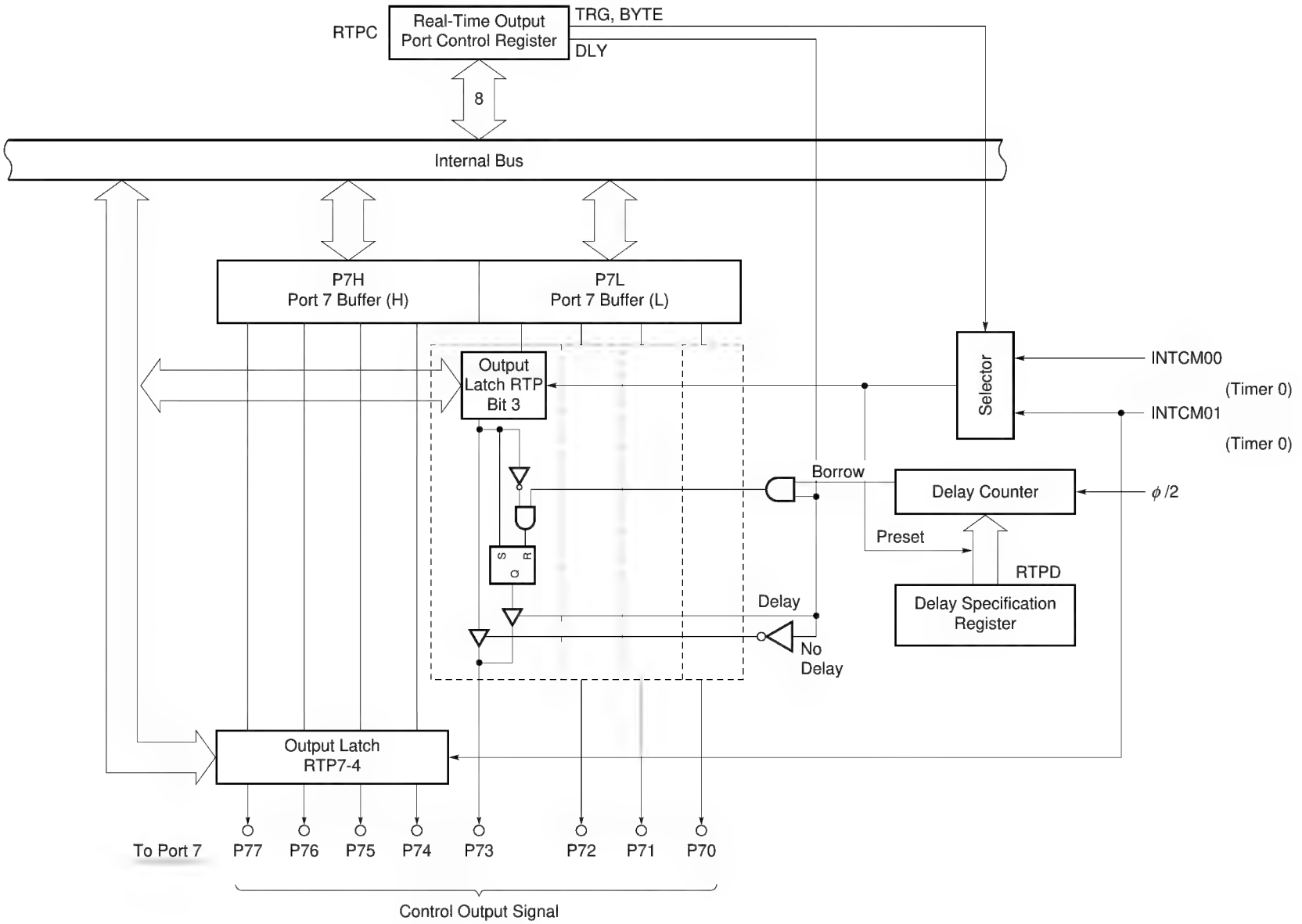
This register is set and used when a mode in which a delay time is inserted into the output from the real-time output port (RTP) to output pin timing .

When the P7L bit is set to "0", "0" is output at the corresponding bit of the output pin after the count clock time delay time set at the real-time output port delay specification register has elapsed after the transfer trigger generation timing. At this time the delay time is counted by the delay counter.

(4) Real-time output port control register (RTPC)

This register sets the real-time output port operation mode. Delay insertion at data output, data transfer to port 7 buffer timing, transfer timing trigger, etc. can be specified.

Figure 10-1. Real-Time Output Port Block Diagram



10.3 Operation

By setting the corresponding bit to the control mode with the port 7 mode control register (PMC7), this pin can be used as the output pin of the real-time output port (RTP). Port 7 (P7), port 7 buffers (P7H, P7L), and the real-time output port can be accessed.

10.3.1 Basic operation

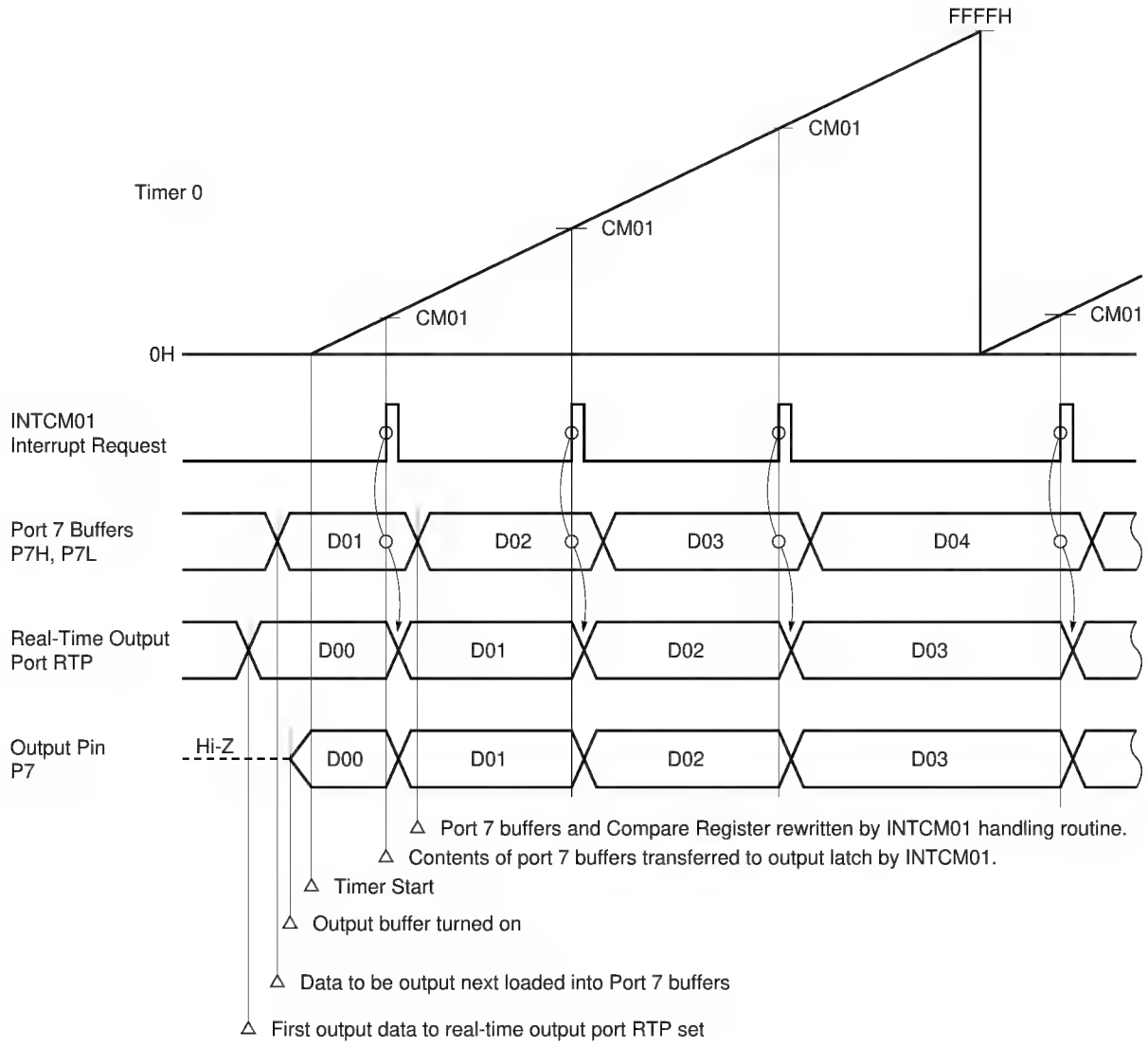
Data output is performed in the following order.

<1> Data is written in the port 7 buffers (P7H, P7L).
<2> The contents of the port 7 buffers are transferred to the RTP when the timer interrupt requests (INTCM00, INTCM01) occur, or when the data is written to the TRG bit in the real-time output port control register (RTPC).
<3> The RTP contents are output to the pin.

The data transfer timing (transfer toggle) of <2> is set by the BYTE bit or TRG bit of the RTPC register. If the timer interrupt request is selected as the transfer trigger, regardless of whether the interrupt request is masked or suspended, data in the buffer register is transferred to the output latch when an interrupt request is generated.

Figure 10-2 shows the real-time output port operations timing when the interrupt request (INTCM01) of timer 0 is made the transfer trigger (BYTE bit=1, TRG bit=0).

Figure 10-2. Real-Time Output Port Operation Timing



10.3.2 Delay function

By setting the real-time output port delay specification register (RTPD), delay time can be added into the output timing for the pin. However, only the P7L buffer can be used.

Only when the port output changes from 1 to 0, the output data change timing can be delayed. The delay time is determined by the delay counter.

(1) Setting

the RTPD register and delay counter become valid when the DLY bit of the RTPC register is set to 1. The delay time is calculated as follows (When RTPD=0, enter 256 for the RTPD setting value.)

$2/\phi \times$ (RTPD set value)

Table 10-1 shows the relationship between the RTPD set value and the delay time.

Table 10-1. RTPD Set Value and Delay Time

RTPD	Delay Time (μs)	
	ϕ = 12.5 MHz	ϕ = 16 MHz
0	40.96	32.0
1	0.16	0.125
2	0.32	0.25
	⋮	⋮
255	40.8	31.875

(ϕ: System clock)

(2) Transfer trigger

The delay specification transfer trigger is as follows according to the setting of the BYTE bit of the RTPC register.

(a) When specified in units of 4 bits (BYTE=0)

- Generation of interrupt request (INTCM00)
- Setting of TRG bit of the RTPC register

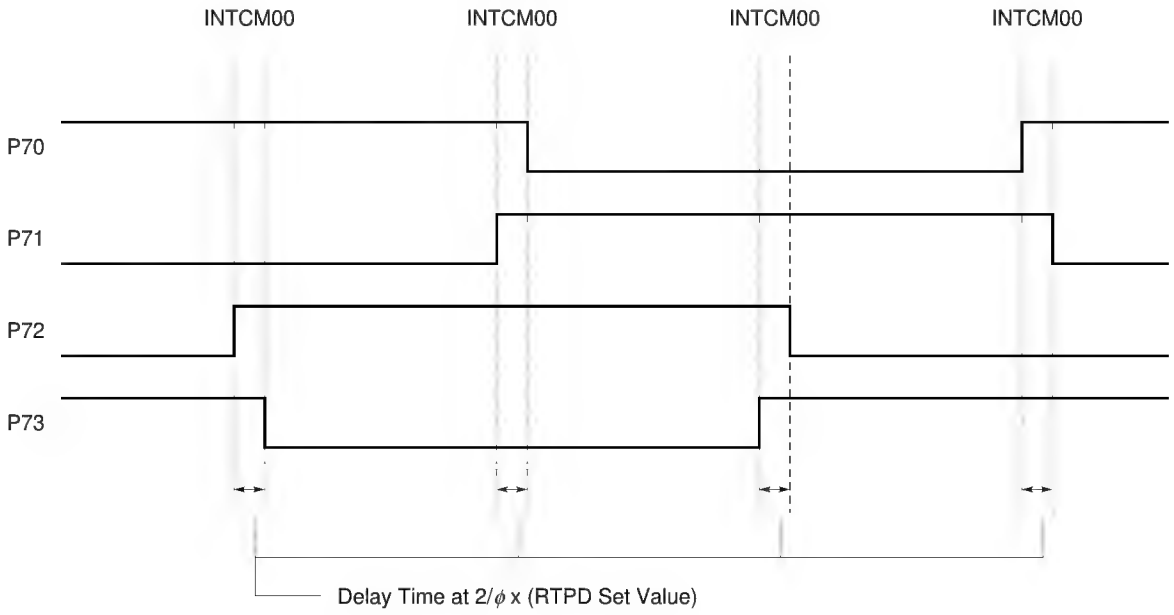
(b) When specified in units of 8 bits (BYTE=1)

- Generation of interrupt request (INTCM01)
- Setting of TRG bit of the RTPC register

(3) Operation

<1> The RTPD register value is set in the delay counter when the transfer triggers are generated (INTCM request or TRG bit setting).
<2> The down count is started (Decreased by 2/ϕ timing).
<3> The value is output to the pin according to the data stored in the P7L buffer at the following timings.
- When the data is 1: Output to the pin as it is.
- When the data is 0: Output to the pin at the timing when the borrow of the delay counter is generated.

Figure 10-3. Real-Time Output Port Output Operation (Delay Mode)



10.3.3 Use with macro service

(1) Setting

The macro service is selected by setting the MS/$\overline{\text{INT}}$ bit of the interrupt request control register.
The operation mode is set to the RTOPTRN mode by the macro service mode register.

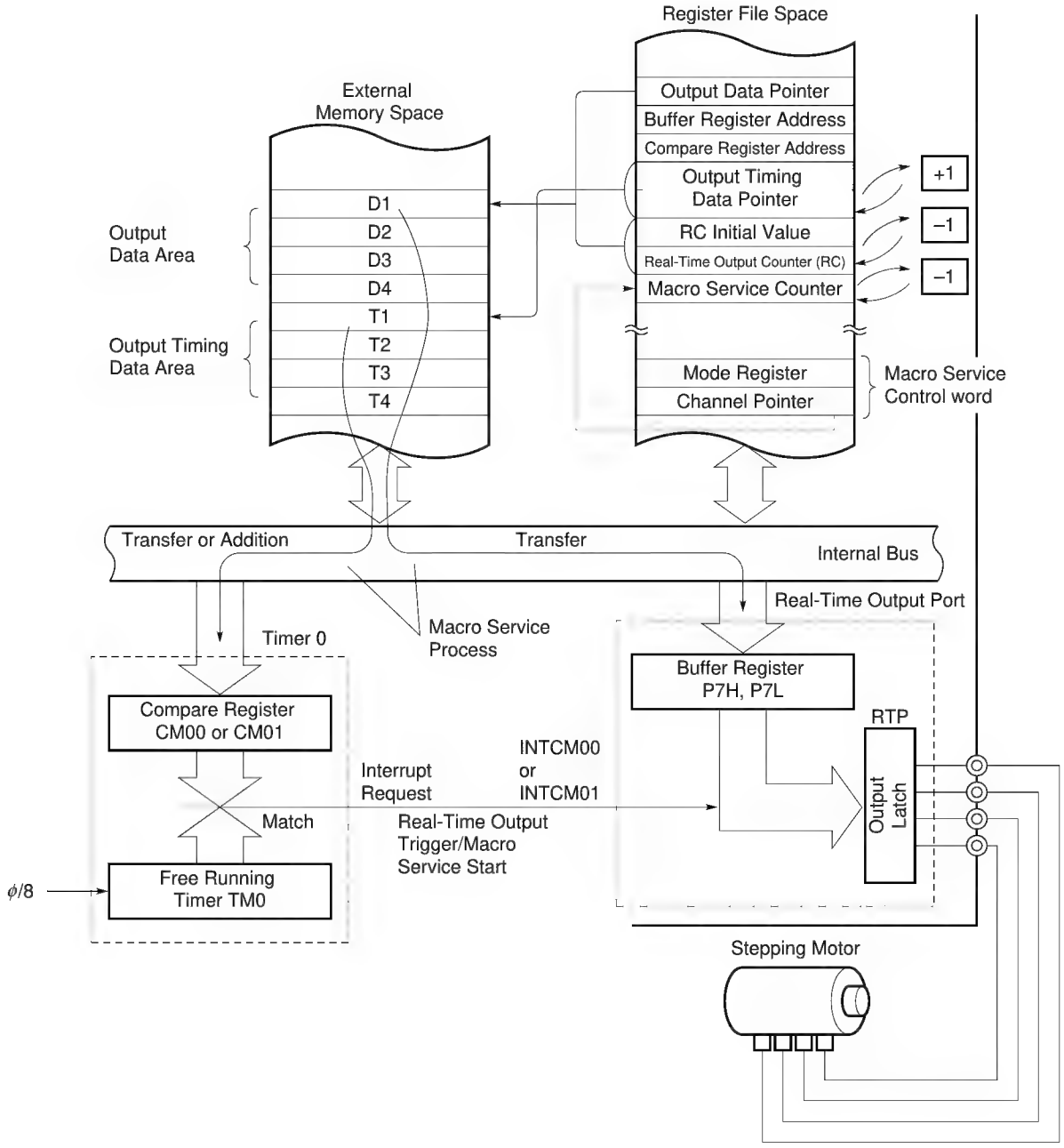
(2) Operation

From two data storage areas preset in the external memory space, updated data is transferred to the real-time output port function buffer registers (P7H and P7L) and compare registers (CM00 and CM01) (See **5.4.3 (6) (f) RTOPTRN (REAL TIME OUTPUT PORT TRANSFER**).

10.3.4 Application example

Figure 10-4 shows an example of the direct control of the output pattern and output interval for the real-time output port.

Figure 10-4. Stepping Motor Control by Real-Time Output Port



10.3.5 Port access operations

Table 10-2 shows the SFR accessed when port 7 is set to an operation mode by the PMC7 register and RTPC register and read/write operations are performed.

If the 8-bit real-time output port is specified (BYTE bit=1), even if either P7H or P7L is specified for access and writing is performed, the 8-bit data is written in the P7 buffer at one time.

Even if either address has been specified to perform reading, P7H and P7L can be read together.

Table 10-2. Port 7 Access Objective

BYTE (RTPC Register)	PMC7n	Operation Mode	Access SFR	Read Access Objective		Write Access Objective	
				High-order 4 bits	Low-order 4 bits	High-order 4 bits	Low-order 4 bits
1/0	0	Port mode	P7	**Note**		Port 7 output latch	
			P7H	P7H	P7L	P7H	–
			P7L	P7H	P7L	–	P7L
			RTP	RTP		RTP	
0	1	4-bit separate real-time port	P7	RTP		P7	
			P7H	P7H	P7L	P7H	–
			P7L	P7H	P7L	–	P7L
			RTP	RTP		RTP	
1	1	8-bit real-time output port	P7	RTP		P7	
			P7H	P7H	P7L	P7H	P7L
			P7L	P7H	P7L	P7H	P7L
			RTP	RTP		RTP	

Note By setting the port read control register (PRDC), the read access object changes according to the bit of the port 7 mode register (PM7) as shown below.

- When PRDC=0
  PM7n=0 (output mode) bit ..... Port 7 output latch value
  PM7n=1 (input mode) bit ..... Port 7 corresponding pin level
- When PRDC=1
  PM7n=0 (output mode) bit ..... Port 7 corresponding pin level
  PM7n=1 (input mode) bit ..... "0"

Remark –: Write access is not performed and RTP, P7H and P7L are not affected.

10.4 Control Registers

10.4.1 Port 7 mode control register (PMC7)

The PMC7 register is an 8-bit register that sets port 7 as a real-time port in bit units. It can be read/written by 8/1 bit memory access. When port 7 is used as a real-time output port, the pertinent bits are set to 1. Bits other than those used as real-time input/output ports can be used as general-purpose ports.

After reset, the value of the PMC7 register becomes 00H and all the bits are set to the port mode.

	7	6	5	4	3	2	1	0
PMC7	PMC77	PMC76	PMC75	PMC74	PMC73	PMC72	PMC71	PMC70

P7 Bits Mode Specification

PMC7n	Mode Specification
0	Input/output port
1	Real-tme output port

Remark The shaded area indicates status after a reset.

10.4.2 Real-time output port control register (RTPC)

The RTPC register is an 8-bit register that specifies the real-time output port operation mode. It can be read/written by 8-bit or 1-bit memory access.

After reset, the value of the RTPC register becomes 40H.

	7	6	5	4	3	2	1	0
RTPC	TRG	BYTE	DLY	0	0	0	0	0

Data output delay specification

DLY	Mode specification
0	No delay mode
1	Delay mode

Transfer timing specification

BYTE	Transfer Timing	
0	P7L	INTCM00 timing
	P7H	INTCM01 timing
1	P7L	INTCM01 timing
	P7H	

Transfer timing trigger specification

TRG	Trigger
0	Timer 0 interrupt request
1	Write to this bit

Remark The shaded area indicates status after a reset.

TRG Transfer timing trigger specification

This bit specifies the trigger source which transfers the value of the port 7 buffers (P7H, P7L) to the real-time output port (RTP).

TRG	Real-Time Output Port Transfer Timing Specification
0	Transfer register contents by timing that activates interrupt request[Note] generated from timer 0
1	Transfer register contents by write to TRG bit timing (1 held after transfer)

Note P7L : INTCM00 (BYTE bit = 0)
INTCM01 (BYTE bit = 1)
P7H : INTCM01

BYTE **Transfer timing specification**

This bit sets the data transfer timing from port 7 buffers (P7H, P7L) to the output port (RTP) (See **Table 10-3**). This bit is valid only when TRG bit = 0.

Table 10-3. Data Transfer Timing to Real-Time Output Port (RTP)

BYTE	TRG	P7H	P7L
0	0	INTCM01 active timing	INTCM00 active timing
0	1	Write 1 to TRG bit timing (1 is also held after transfer)	
1	0	INTCM01 active timing	
1	1	Write 1 to TRG bit timing (1 is also held after transfer)	

DLY **Data output delay specification**

This bit specifies the mode which inserts a delay time until P7L "0" data is output to the output pin when a transfer trigger was generated.

When the transfer data is "1", a delay is not inserted into the output timing. This bit is valid only for the low-order side P7L of the port 7 buffers. The pin output always changes simultaneously with data high-order side P7H data transfer.

DLY	Mode	Data Output Timing from Pin	
0	No Delay	The data of each bit of P7L is output from the output pin in synchronization with the transfer to output port (RTP) trigger generation timing.	
1	Delay	Output bit data = 0	Output pin after insertion of the delay set at the RTPD register from the transfer to output port (RTP) trigger generation timing.
		Output bit data = 1	Output from pin in synchronization with the transfer to output port (RTP) trigger generation timing.

10.4.3 Real-time output port delay specification register (RTPD)

The RTPD register is an 8-bit register that specifies the delay timing until "0" data of the port 7 buffer (P7L) is output from the output pin when a transfer trigger was generated. It can be read/written by 8/1 bit memory access.

The value of the RTPD register becomes undefined at reset input.

Setting of this register is valid only when the DLY bit of the real-time output port control register (RTPC) is 1.

10.4.4 Delay counter

This is an 8-bit counter that counts the delay time until the "0" data of the port 7 buffer (P7L) is output at the output pin after a transfer trigger was generated.

It is valid only when the RTPC register DLY bit (bit 5) = 1.

This counter cannot be read/written by instruction.

Caution When "delay mode" is switched to "no delay mode" during the delay counter operation, the output status of the P70 to P73 pins are not changed until the timing for transferring the P7L buffer value to the real-time output port is generated.
To output the RTP0 to RTP3 values to P70 to P73, the manipulation of the TRG bit of the RTPC register must be performed.

CHAPTER 11 PWM OUTPUT FUNCTION

The V55PI has an 8-bit accuracy PWM (Pulse Width Modulation) signal output function.

This PWM output can be used as a digital/analog conversion output by connecting an external low-pass filter, etc. and is ideal as the actuator control signal for a motor, etc.

11.1 Features

- PWM output pulse active level can be selected.
- Frequency: 25 MHz (system clock ϕ = 12.5 MHz)
  → PWM period: 40.96 μs
  : 32 MHz (system clock ϕ = 16 MHz)
  → PWM cycle: 32.00 μs
- Output pulse width (duty): 0, 1/256, ..., 255/256
  → Resolution: 160 ns (system clock ϕ = 12.5 MHz)
  125 ns (system clock ϕ = 16 MHz)

11.2 Configuration

The functions of each block are as follows. The PWM unit configuration is shown in **Figure 11-1**.

(1) PWM register

Sets the PWM output duty ratio (active level width). Can be rewritten using any timing.
It is not affected by reset input.

(2) PWM control register

Controls the settings and output of the PWM output active level.

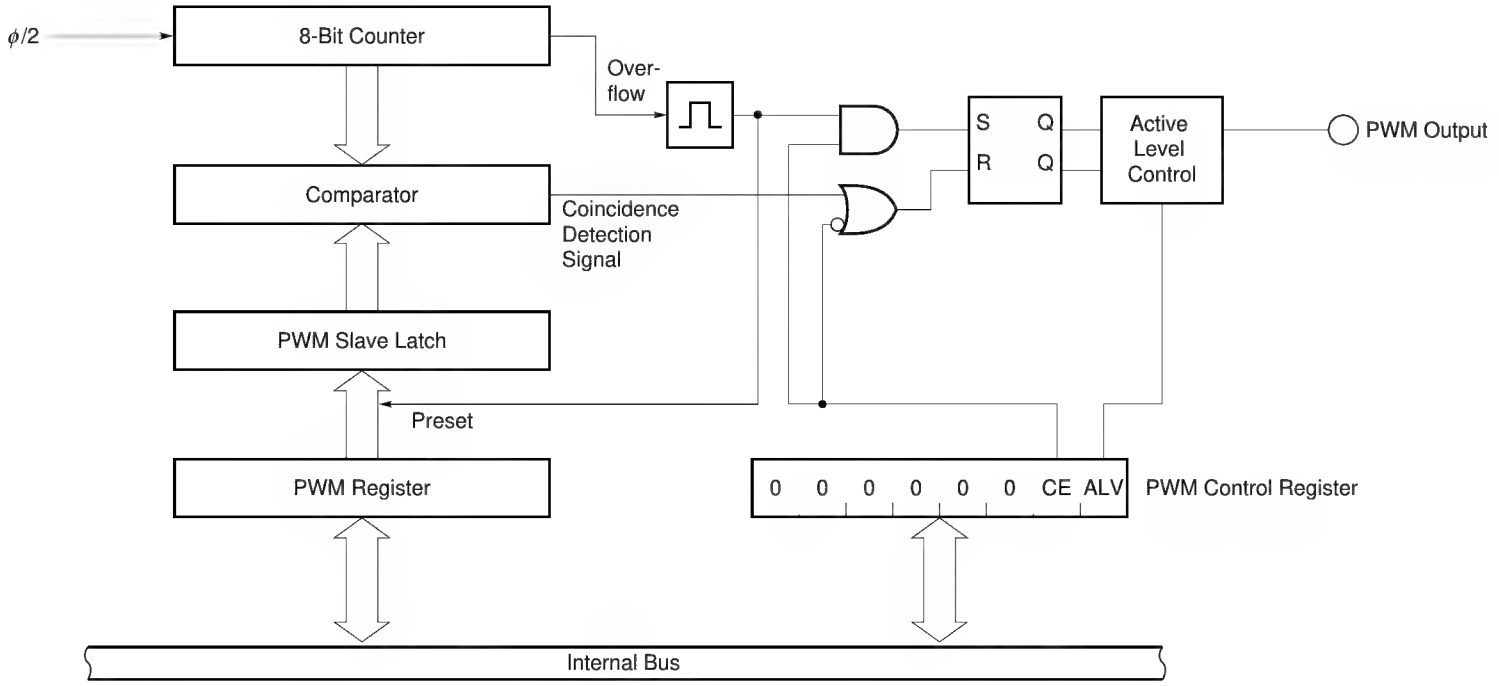
(3) 8-bit counter

Counts up according to the ϕ/2 clock. It is set to 00H by reset input.

(4) PWM slave latch

This register sets the active level width.
It is downloaded from the PWM register when the 8-bit counter overflow is detected.

Figure 11-1. PWM Unit Block Diagram



11.3 Operation

11.3.1 Setting

Set and specify the corresponding bit (1) of the port 2 mode control register (PMC2) for the control mode.

The operation thereafter is controlled by the CE bit of the PWM control register. When the CE bit is made 1, the PWM pin level becomes active by the tap output from 2^9 of the next time base counter (TBC) and the PWM register value is transferred to the PWM slave latch. The 8-bit counter continues to count up according to the $\phi/2$ count clock (See **Figure 11-3**).

11.3.2 Basic operation

The following basic operation is repeated until CE becomes 0.

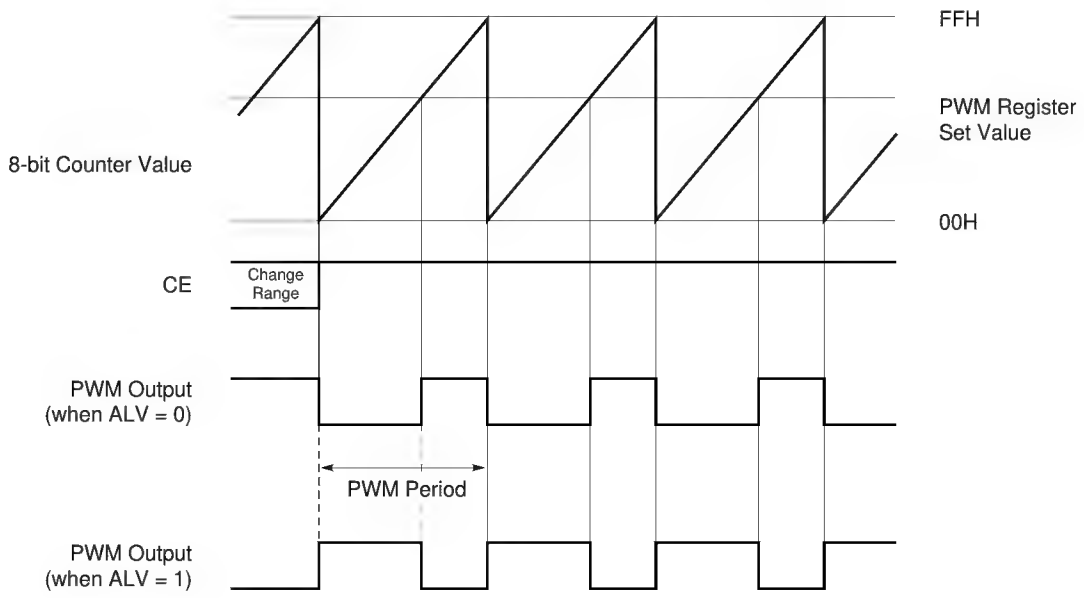
The 8-bit counter continues count up regardless of the value of the CE bit.

<1> The active level width of the PWM output is set for the PWM register.

<2> Data is transferred to the PWM slave latch when the 8-bit counter overflow is detected. The PWM output is set and made an active level.

<3> When the 8-bit counter value and PWM slave latch value coincide, the PWM output is made inactive.

Figure 11-2. PWM Output Operation Timing



11.3.3 PWM output operations

The following operations are valid during PWM output.

- By rewriting the PWM register contents at a certain timing, the active level width of the PWM output can be changed (See **11.4.2 PWM register (PWM)**).
  As the PWM register value is preset in the PWM slave latch when the 8-bit counter overflow is detected, the currently output active level width is not affected.
- When the CE bit of the PWM control register is reset (0), the PWM output is made inactive immediately even during PWM output (See **Figure 11-4**).
- The active level of the PWM output can be switched using the ALV bit of the PWM control register. In this case, the PWM output will be switched even during the PWM period (See **Figure 11-5**).

Figure 11-3. PWM Output Operation Timing (CE = 0 → 1)

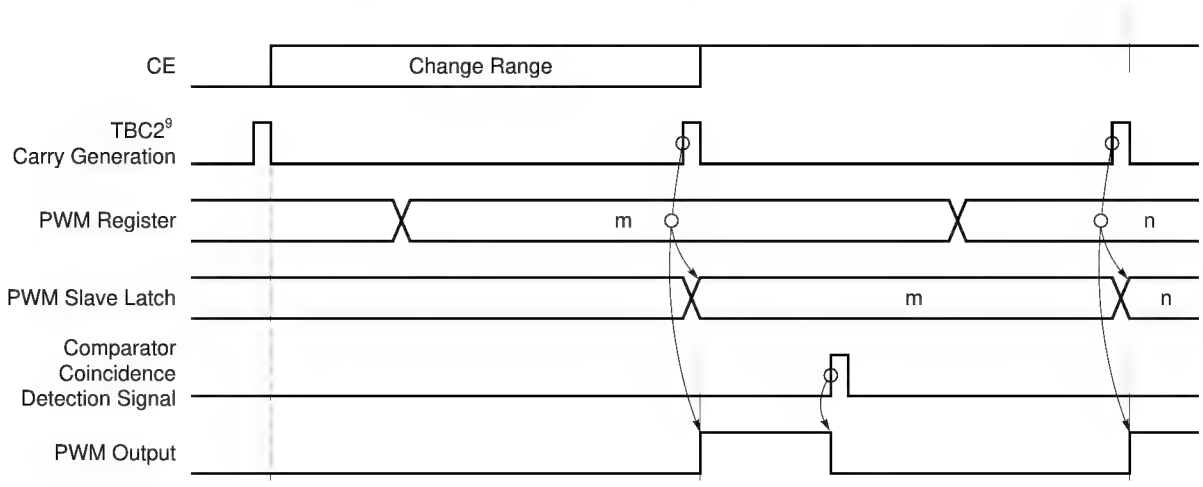


Figure 11-4. PWM Output Operation Timing (CE = 1 → 0)

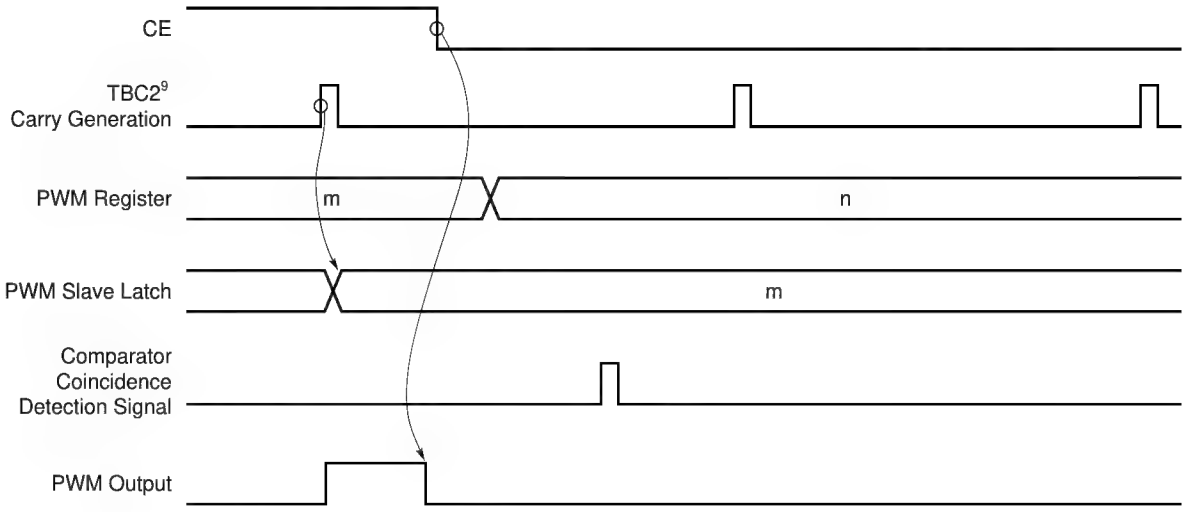
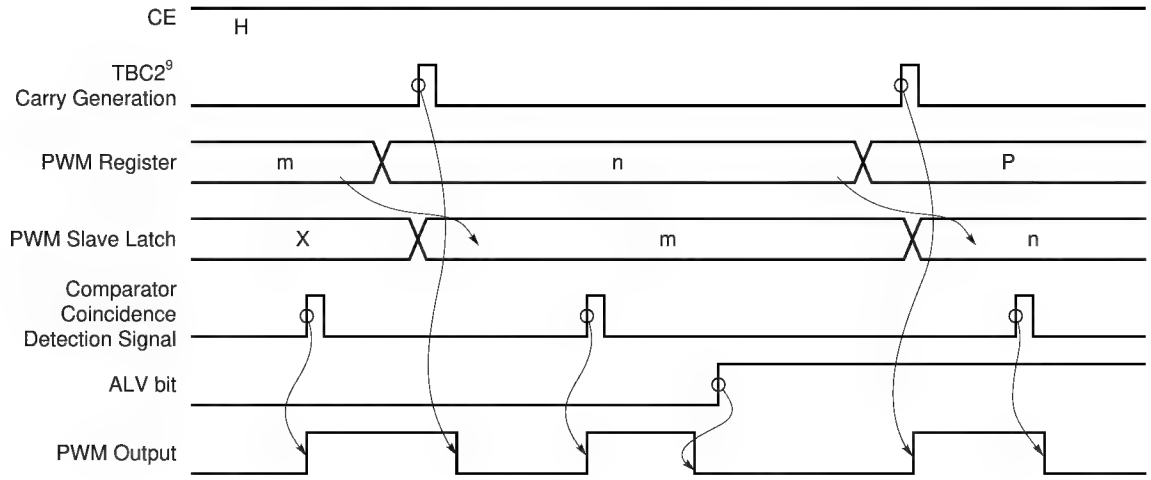


Figure 11-5. PWM Output Operation Timing (When the ALV bit is changed)



11.4 Control Registers

11.4.1 PWM control register (PWMC)

The PWMC register is an 8-bit register that controls the PWM register (PWM) operation mode and PWM output. After reset, the value of the PWMC register becomes 00H.

	7	6	5	4	3	2	1	0
PWMC	0	0	0	0	0	0	CE	ALV

PWM Output Active Level Specification

ALV[Note]	Active Level
0	Low level
1	High level

PWM Operation Specification

CE	PWM Operation
0	Stop operation (PWM output not changed by $\overline{ALV}$ level)
1	Enable operation (PWM output changed)

Note When ALV was switched, the output value changes immediately after switching.

Remark The shaded area indicates status after a reset

11.4.2 PWM register (PWM)

The PWM register is an 8-bit register that sets the PWM output active level width. The PWM output active level width tw is determined by the following expression by setting to the PWM register.

tw = PWM register value × tCYC

tCYC is the internal system clock cycle time and is $2/\phi$.

Table 11-1. PWM Register Value and Duty Ratio

PWM Register Value	Duty
00H	0[Note]
01H	1/256
⋮	⋮
FFH	255/256

Note The duty can be made 100% by inverting the ALV specification and resetting (0) the CE bit.

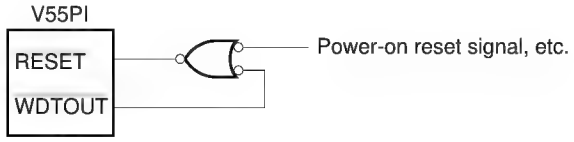
CHAPTER 12 WATCHDOG TIMER FUNCTION

The watchdog timer is a function for preventing program runaway and deadlock.

12.1 Features

- Three overflow times can be set (10.4, 41.9, 167.7 [ms]: system clock ϕ = 12.5 MHz)
  (8.1, 32.7, 131.0 [ms]: system clock ϕ = 16 MHz)
- Output pin ($\overline{\text{WDTOUT}}$ pin) connectable to $\overline{\text{RESET}}$ pin

★ **Figure 12-1. $\overline{\text{RESET}}$ Pin Connection Example**



12.2 Configuration and Operation

That the program or system is operating normally can be verified by the fact that a watchdog timer interrupt is not generated. To use the watchdog timer, an instruction (RSTWDT instruction) that clears (starts counting) the watchdog timer at a program execution fixed interval or at the top of a subroutine, etc. is necessary.

If an instruction that clears the watchdog timer is not executed within the set time and the watchdog timer overflows, a watchdog timer interrupt (INTWDT) is generated and low level is output at the $\overline{\text{WDTOUT}}$ pin to indicate a program error.

The watchdog timer block diagram is shown in **Figure 12-1**.

(1) Watchdog timer

The watchdog timer also serves as the main clock oscillator oscillation stabilization time confirmation timer. When system reset is input or the V55PI enters the STOP mode, the counter value is cleared to 00H.

Once the watchdog timer is started, writing to the watchdog timer mode register (WDM) is disabled by the hardware and the software cannot change the interval timing, stop the watchdog timer, etc. until the watchdog timer overflows. (However, the watchdog timer can be cleared and the WDM register can be read at any time.)

In the standby mode, the watchdog timer operates as follows.

(STOP mode)

Counting is stopped. When the STOP mode is released by NMI input, counting is started after the end of effective level input to the NMI pin. When the watchdog timer overflows, internal clock supply resumes. INTWDT is not generated at this time. (If the main clock oscillator oscillation fails to stabilize within the time up to watchdog timer overflow after the STOP mode is released, the system clock will become unstable and erroneous operation may occur.)

(HALT MODE)

Counting continues. Therefore, the HALT mode is released by INTWDT.

(HOLD mode)

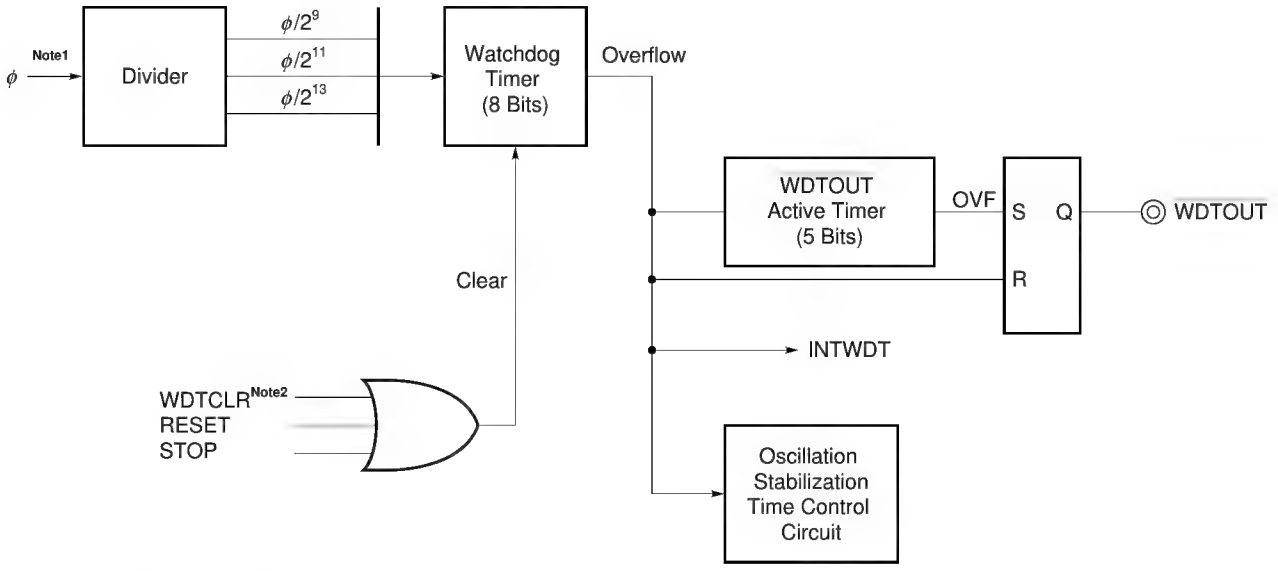
When the CPU is placed into the HOLD mode by external request, the contents of the counter are held and counting stops. After the HOLD mode is released, counting is resumed automatically. HOLD transition by internal DMA does not stop the count.

(2) $\overline{\text{WDTOUT}}$ active timer

This timer holds the signal that is output when a watchdog timer interrupt request is generated at active low for 32 clocks. System reset can be applied by watchdog timer overflow by connecting the $\overline{\text{WDTOUT}}$ and $\overline{\text{RESET}}$ pins externally.

Caution Immediately after power-on reset, the $\overline{\text{WDTOUT}}$ pin output becomes undefined. At this time, this pin becomes high level for up to 32 clocks after reset input.

Figure 12-2. Watchdog Timer Block Diagram



Notes 1. ϕ : System clock

2. WDTCLR: Watchdog timer clear by instruction

Application using watchdog timer

In the initial stages of program development, the watchdog timer is not used and debugging including the watchdog timer is performed after rough program debugging is complete.

When the response time after an application system error occurred is important, a mode with a small overflow time is selected. When not used as a watchdog timer, the watchdog timer function can be used as a nonmaskable time base interrupt.

12.3 Control Registers

12.3.1 Watchdog timer mode register (WDM)

The WDM register is an 8-bit register that controls the watchdog timer operation.

The WDM register can be written by dedicated instruction only so that its contents are not rewritten erroneously by program runaway.

RSTWDT imm8, imm8'

This dedicated instruction has a special code structure (4 bytes). Writing is not performed if the opcodes imm8 and imm8' are not mutual complements.

When the operands above are not complements, an exception trap interrupt is generated. (See section **5.8 Exception Trap**.)

After system reset ($\overline{\text{RESET}}$ input), once the watchdog timer is started by setting the RUN bit of the WDM register to 1, it can only be stopped by system reset only.

Attempts to rewrite the RUN bit to 1 will be invalid and the WDM register contents will not change. However, the WDM register can be read anytime using the data transfer instruction.

The watchdog timer can be cleared at anytime by executing the dedicated instruction (RSTWDT). Set so that the operation codes (imm8 and imm8') of this instruction mutual complements (See **V55PI User's Manual–Instructions**).

A value after a reset is 00H.

	7	6	5	4	3	2	1	0
WDM	RUN	0	0	0	0	WDI2	WDI1	0

Overflow Time Setting

WDI2	WDI1	Count Clock	Overflow Time	
			ϕ = 12.5 MHz	ϕ = 16 MHz
0	0	$\phi/2^9$	10.4 ms[Note]	8.1 ms[Note]
0	1	$\phi/2^{11}$	41.9 ms	32.7 ms
1	0	$\phi/2^{13}$	167.7 ms	131.0 ms
1	1	Reserved	Setting prohibited	Setting prohibited

(ϕ : System clock)

Watchdog Timer Specification

RUN	Operation Specification
0	Stop watchdog timer
1	Clear watchdog timer and start count

Note Count time as oscillation stabilization time at STOP mode release.

Remark The shaded area indicates status after a reset.

CHAPTER 13 A/D CONVERTER FUNCTION

The V55PI has an on-chip high-speed, high-accuracy 8-bit analog/digital (A/D) converter with four analog inputs (ANI0-ANI3). The A/D converter uses a successive approximation method and has four A/D conversion result registers (ADCR0-ADCR3) that hold the converted result.

13.1 Features

- Four on-chip 8-bit A/D converted result registers.
- Analog input pins (ANI0-ANI3): 4
- A/D converter conversion operation modes: 2
  - Scan mode : Multiple analog inputs are sequentially selected and converted.
  - Select mode : Analog input is fixed at one pin and converted continuously.
- Conversion start method: 2
  - Hardware start : Start by trigger input (INTP4)
  - Software start : Start by A/D converter mode register (ADM) bit setting
- Conversion end interrupt request (INTAD) generation

13.2 Configuration

Four analog inputs (ANI0 to ANI3) are provided and these are selected by the A/D converter mode register (ADM). The selected analog input becomes one of the inputs of the voltage comparator.

The functions of each block are as follows.

(1) Input circuit

The analog inputs are selected by the specification of the A/D converter mode register (ADM) and transferred to the sampling and hold circuit according to the operation mode.

(2) Sampling and hold circuit

Samples the analog inputs transferred sequentially one by one and maintains those being A/D converted.

(3) Voltage comparator

Compares the analog input voltage and the voltage tap voltage of the series resistance string.

(4) Series resistance string

Generates the voltage used for comparing with the analog input voltage.

The series resistance string is connected between the A/D converter reference voltage pin (AV_{REF}) and A/D converter GND pin (AV_{SS}), and consists of 255 equivalent resistors and two resistors of 1/2 its value to make the voltage across two pins 256 equivalent voltage steps.

The voltage tap of the series resistance string is selected by the tap decoder controlled by the SAR register.

(5) 8-bit SAR register (successive approximation register)

The SAR register is set and compared one bit at a time from the most significant bit (MSB) so that the value of the series resistance string voltage taps coincides with the voltage value of the analog input.

After the setting is completed to the last significant bit (LSB) (A/D conversion end), the SAR register contents (conversion results) is kept in the A/D conversion result register.

(6) A/D conversion result register (ADCR0 to ADCR3)

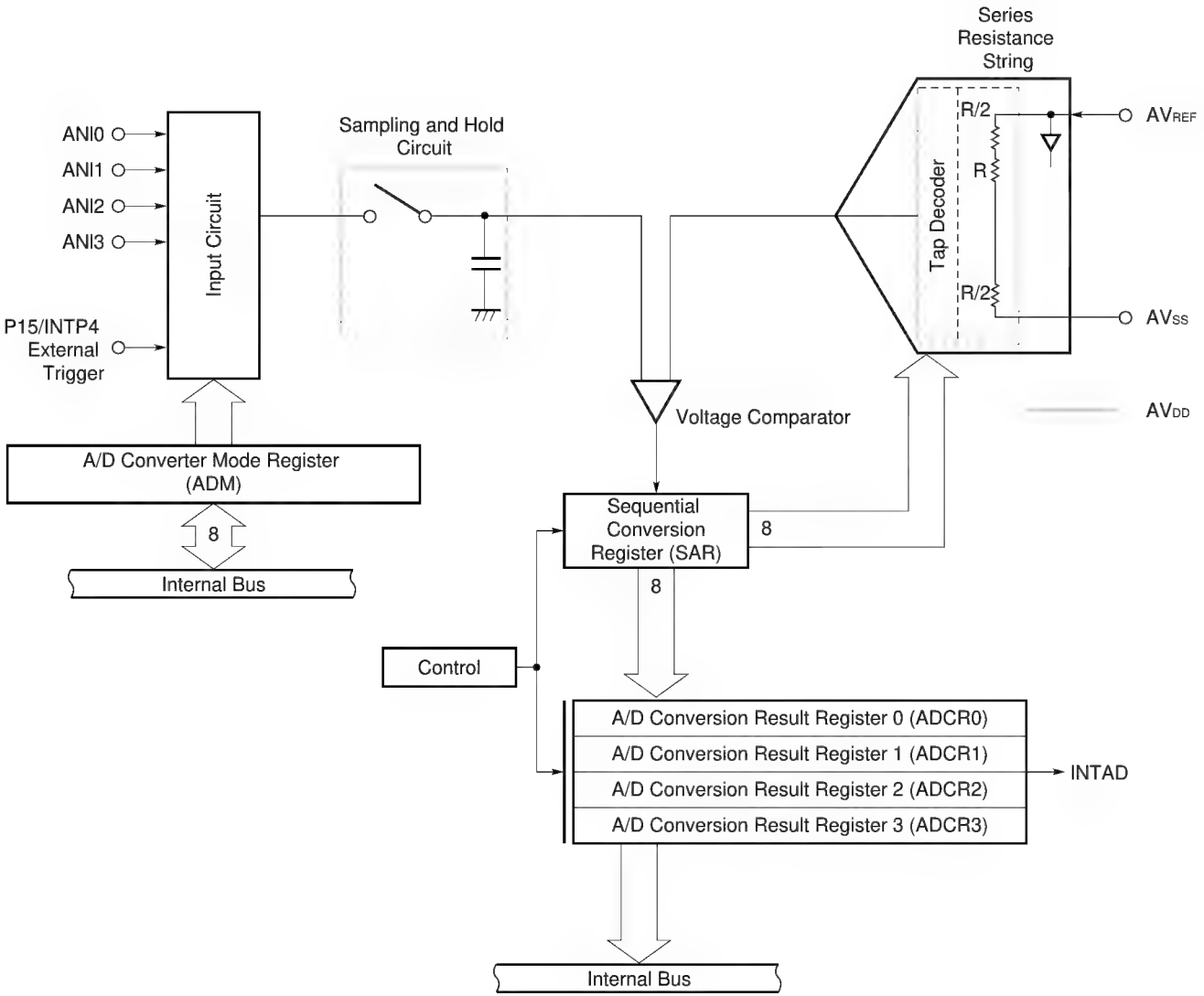
This 8-bit register keeps the A/D conversion result. When the A/D conversion ends, the conversion results is loaded from the SAR register. When the data is loaded, the conversion end interrupt request (INTAD) is generated. The status becomes undefined when reset is performed.

The four analog inputs (ANI0 to ANI3) have a 1 to 1 correspondence with the four registers (ADCR0 to ADCR3) as shown below.

Analog Input	ADCR Register
ANI0	ADCR0
ANI1	ADCR1
ANI2	ADCR2
ANI3	ADCR3

Caution Be sure that a voltage outside the AV_{SS}-AV_{REF} range is not applied to the ANI0-ANI3 pins whether A/D conversion is used or not used.

Figure 13-1. A/D Converter Block Diagram



13.3 Operation

The A/D converter selects the select mode or the scan mode. The following describes common basic operations of the two modes, and each mode.

13.3.1 Basic operations of A/D converter

(1) A/D conversion operation flow

A/D conversion is performed as follows.

(a) Analog input selection and operation mode specification are set by the A/D converter mode register (ADM).

(b) Bit 7 (CS) of the ADM register is set (1) and A/D conversion is started.

(c) The most significant bit (bit 7) of the SAR register is set automatically (1) when conversion starts.

(d) When bit 7 of the SAR register is set, the tap decoder sets the voltage tap of the series resistance string to $\frac{255}{512}AV_{REF}$ ($\cong \frac{1}{2}AV_{REF}$).

(e) Voltage difference between the voltage tap of the series resistance string and the analog input is compared by the voltage comparator. If the analog input is greater than (1/2) AV_{REF}, the MSB of the SAR register is set as it is.

If the analog input is smaller than (1/2) AV_{REF}, the MSB is reset.

(f) Next, bit 6 of the SAR register is automatically set (1) and the next comparison is performed. The series resistance string voltage tap shown below is selected according to the bit 7 value already set.

- bit 7 = 1 ..... $\frac{383}{512}AV_{REF} \cong \frac{3}{4}AV_{REF}$
- bit 7 = 0 ..... $\frac{127}{512}AV_{REF} \cong \frac{1}{4}AV_{REF}$

This voltage tap is compared with the analog input voltage and bit 6 of the SAR register is manipulated as follows according to the results.

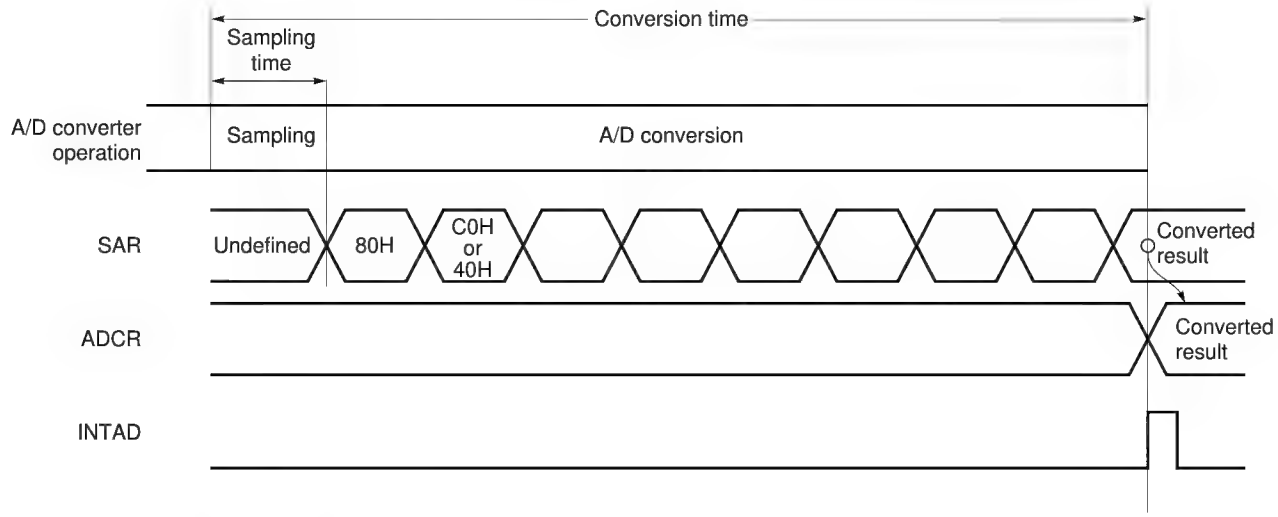
- Analog input voltage ≥ voltage tap: Bit 6 = 1
- Analog input voltage < voltage tap: Bit 6 = 0

(g) This comparison is continued to most significant bit (bit 0) of the SAR register (binary search method).

(h) When the 8-bit comparison ends, the valid digital result remains in the SAR register, the value is transferred to the A/D conversion result register (ADCR) and latched.

At the same time, the A/D conversion end interrupt request (INTAD) is generated.

Figure 13-2. A/D Converter Basic Operation



(2) A/D conversion operation control

A/D conversion is started by the following two methods.

Either one method is selected by the specification of the TRG bit of the ADM register.

- **Start by CS bit of the ADM register (When TRG = 0)**

  Conversion starts when the CS bit is made 1.

  The A/D conversion of the channel specified by bits 0 to 2 (MS, AN0 and AN1) of the ADM register is repeatedly executed. When one conversion ends, the next one is performed.

- **Start by input of an external pin trigger INTP4 (When TRG = 1)**

  The operation stops in a wait status for effective edge input, and the conversion starts when the trigger is input. Once conversion is started by inputting the external trigger, it will not stop even if the TRG bit is reset (1) or the effective edge is input to INTP4. Only the conversion sequence will be initialized. Therefore, to stop operations and set the wait status for start of conversion by the external trigger input again, set as follows.

  <1> CS ← 0, TRG ← 1 (conversion operation stop)

  <2> CS ← 1, TRG ← 1

A/D conversion is stopped by resetting the CS bit (0). Until then, conversion is performed continuously. It can also be stopped by resetting the CS bit.

When writing is performed to the ADM register during A/D conversion, the conversion sequence is initialized and continued.

The conversion time can be set using the FR bit of the ADM register. This is the time from the start of conversion to the latching of the conversion result to the ADCR register.

The ADCR register is not affected by reset.

(3) A/D conversion time

The time required for A/D conversion is determined by the internal system clock frequency (ϕ) and FR bit of the ADM register.

The A/D conversion time includes the time required for one A/D conversion and also the sampling time.

Table 13-1 shows the value.

Table 13-1. A/D Conversion Time

System Clock (ϕ) Range	FR Bit	Conversion Time	Sampling Time
8 MHz < ϕ < 16 MHz	0	160/ϕ (10 μs to 20 μs)	32/ϕ (3 μs to 6 μs)
4 MHz < $\phi \leq$ 8 MHz	1	120/ϕ (15 μs to 30 μs)	24/ϕ (2 μs to 4 μs)

13.3.2 Select mode

(1) Setting

Set bit 0 (MS) of the ADM register (1).

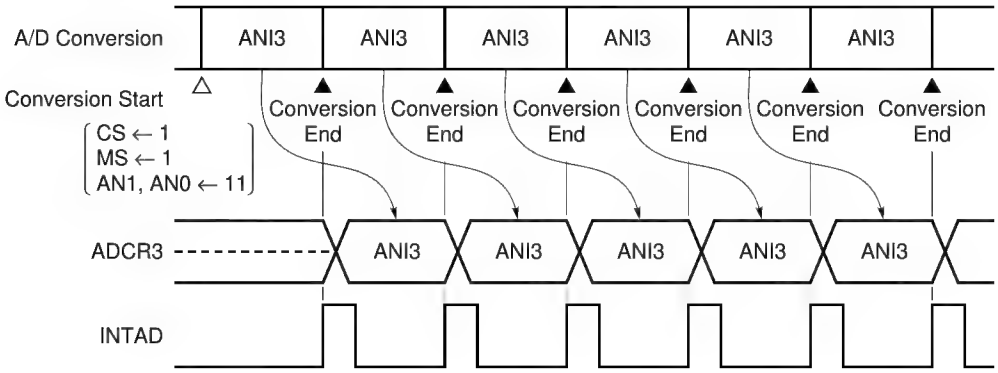
(2) Operation

One analog input is specified by bits 1 and 2 (AN0, AN1) of the ADM register and the A/D conversion of the specified analog input pin is started.

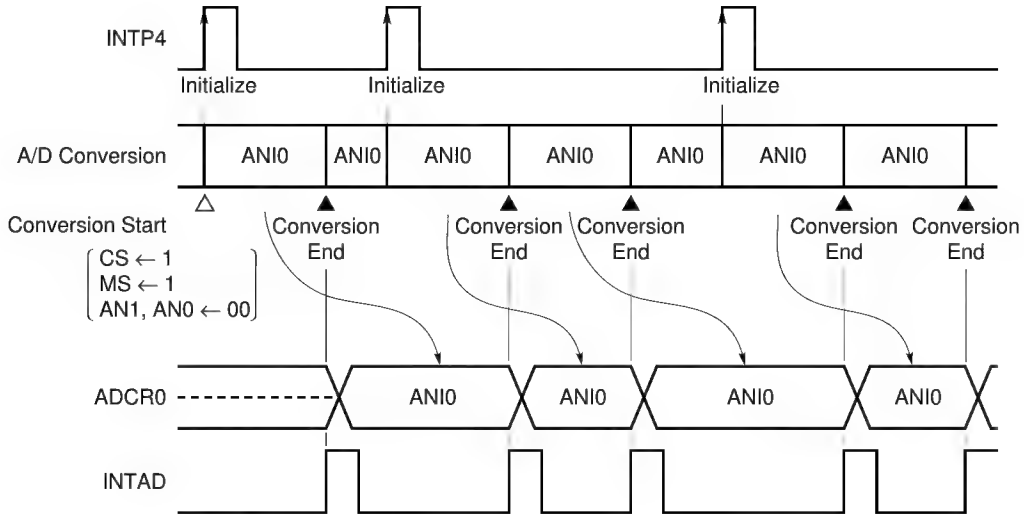
When the conversion of the selected analog input is ended, the conversion results is stored in the A/D conversion result register (ADCR) with 1 to 1 correspondence with the analog input, and interrupt (INTAD) is generated. Regardless of whether the interrupt request has been acknowledged, the conversion of the same analog input is started again. This operation is continued until the ADM register is rewritten.

Figure 13-3. Select Mode Operation Timing

(a) TRG bit ← 0



(b) TRG bit ← 1



13.3.3 Scan mode

(1) Setting

Reset bit 0 (MS) of the ADM register (0).

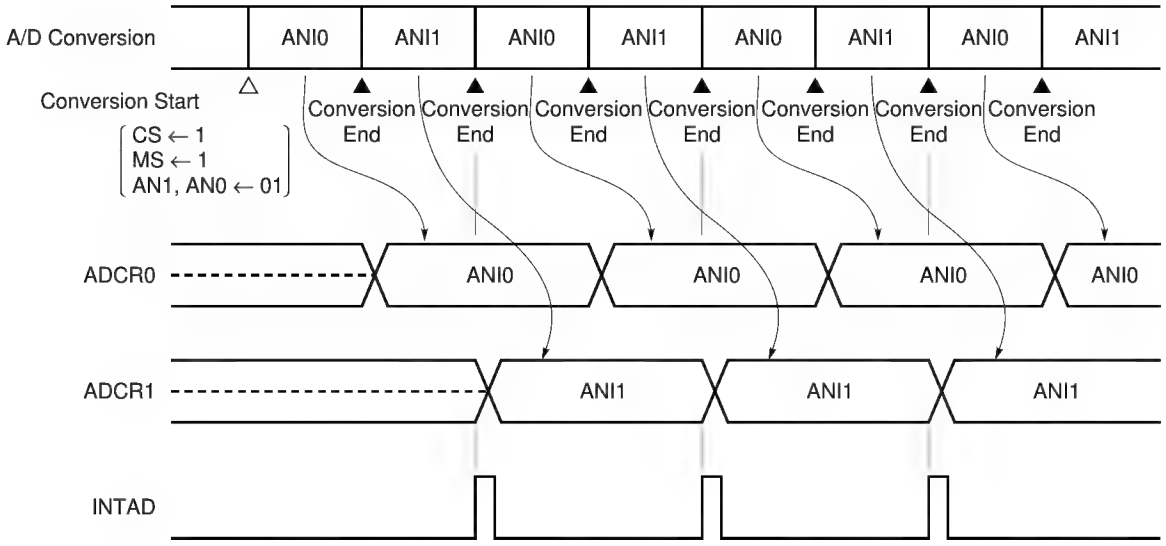
(2) Operation

Inputs from the analog input pins specified by bits 1 and 2 (AN0, AN1) of the ADM register are selected and converted sequentially.

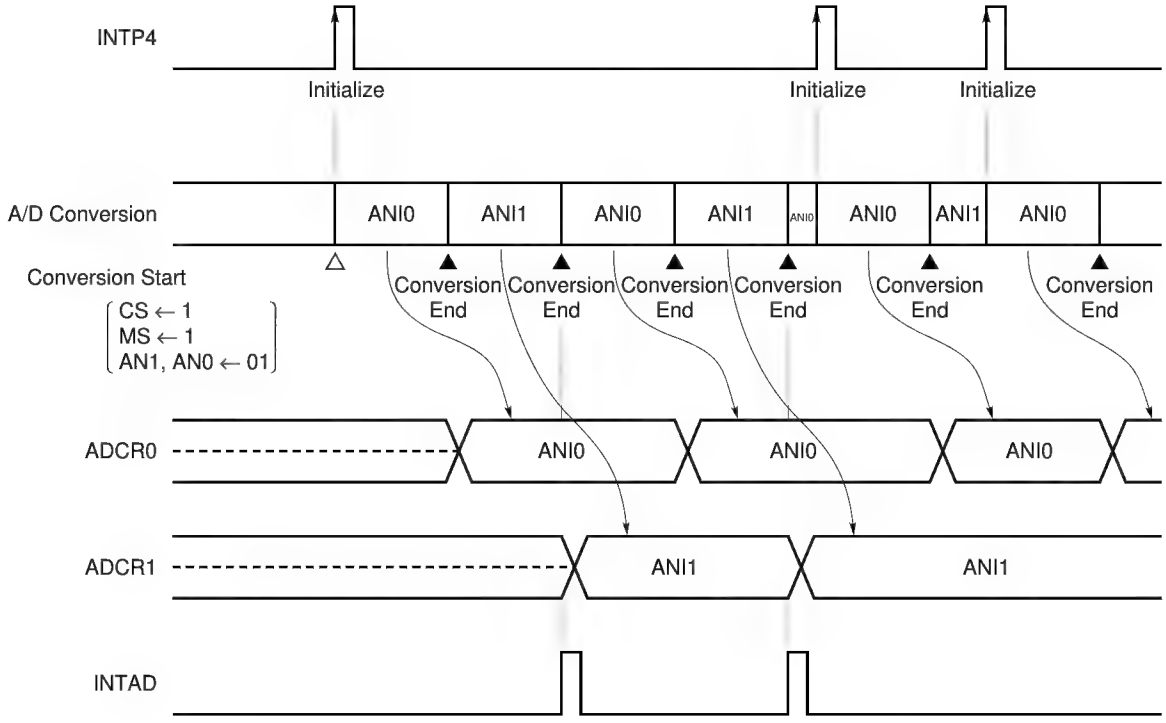
For example, when AN1 and AN0 are 01, ANI0 and ANI1 are repeatedly scanned (ANI0 → ANI1 → ANI0 → ANI1). The conversion result is stored in the ADCR register corresponding 1 to 1 to the analog inputs. Channels are converted sequentially from channel 0 and when all input channels specified by the ADM register have been converted, the interrupt request (INTAD) is generated and a conversion is started from channel 0 again.

Figure 13-4. Scan Mode Operation Timing

(a) TRG bit ← 0



(b) TRG bit ← 1



13.4 Control Registers

13.4.1 A/D converter mode register (ADM)

The ADM register controls the A/D converter operation mode.

After reset, the value of the ADM register becomes 00H.

	7	6	5	4	3	2	1	0
ADM	CS	TRG	0	FR	0	AN1	AN0	MS

Conversion Operation Mode Specification

MS	Conversion Operation Mode
0	Scan mode
1	Select mode

Analog Input Specification

AN1	AN0	Analog Input	
		Scan mode	Select mode
0	0	Scan ANI0 input	Select ANI0 input
0	1	Scan AINI0, ANI1 inputs	Select ANI1 input
1	0	Scan AINI0, ANI1, ANI2 inputs	Select ANI2 input
1	1	Scan ANI0, ANI1, ANI2, ANI3 inputs	Select ANI3 input

Conversion Time Rewriting

FR	Conversion Time
0	160 clock (when 8 MHz < ϕ < 16 MHz)
1	120 clock (when 4 MHz < $\phi \leq$ 8 MHz)

External Pin Trigger Control

TRG	Trigger Control
0	Disable external pin trigger
1	Enable external pin trigger

A/D Conversion Control

CS	Conversion Operation
0	Stop A/D conversion
1	Start A/D conversion

Remark The shaded area indicates status after a reset.

FR Conversion time switching

Control bit used so that the A/D conversion time does not change considerably even if the oscillation frequency changes. The conversion speed of one conversion is determined by the internal system clock ϕ and the FR bit.

When 4 MHz < $\phi \leq$ 8 MHz, set FR = 1

CS A/D conversion control

When CS = 0 is set, all conversion operations are stopped even if conversion is being performed. At this time, the ADCR register is not updated and an interrupt request (INTAD) is not generated.

13.4.2 A/D conversion result register (ADCR0-ADCR3)

The A/D conversion results register is a 8-bit register storing the A/D conversion result.

13.5 Interrupt Request

An A/D converter interrupt request (INTAD) is generated when the converted result of the selected analog input was stored to the ADCR register. The interrupt request control register controls the interrupt mode.

For a description of the interrupt control register, see section **5.5.3 Interrupt request control register (IC9 to 14, 16 to 32, 36 and 37)**.

13.6 Caution

13.6.1 Executing the STOP instruction

When setting the STOP mode while the A/D converter is operating, always stop the A/D converter conversion operations. This is to prevent the sampling gate from being left open when the A/D converter is stopped during sampling operations. By setting the CS bit of the A/D converter mode register to 0, all conversion operations will be stopped even if conversion is being performed.

13.6.2 A/D conversion sequence initialization

The A/D conversion sequence is initialized by effective edge input to the external trigger pin and by resetting (1 → 1) of the TRG bit of the A/D converter mode register. However, set the period of the effective edge input to the external longer than one conversion operation period because conversion is not performed when the A/D conversion is performed by repeated effective edge input to the external pin, etc., if the effective edge input period is shorter than one conversion sequence period.

13.6.3 Loss of conversion result

If the A/D conversion result is read by vector interrupt in the select mode of the A/D converter, the conversion results may be lost if the A/D conversion end interrupt is kept for a long time (160 clocks when the FR bit is 0 and above 120 clocks when the FR bit is 1) by other interrupt processes. As it is difficult to detect this loss of conversion results with the software, take the following measures to prevent such losses.

- Make the time of other interrupt processes sufficiently short compared to the A/D conversion time.
- Using multiple interrupt, acknowledge the A/D conversion end interrupt even if other interrupt processes are performed.
- Perform A/D conversion end interrupt using macro service.

CHAPTER 14 PORT FUNCTIONS

The V55PI pins, except for some, are used as port pins and control pins. For a detailed description of each pin, see section **2.1 List of Pin Functions**.

14.1 Features

- Input port, output port and control function selectable for each pin.
- Dedicated I/O pins: 8
- External interrupt request input/input port pins: 7
- Analog input/input port pins: 4
- Other I/O port pins that can be switched to controlling function pins: 34

14.2 Configuration

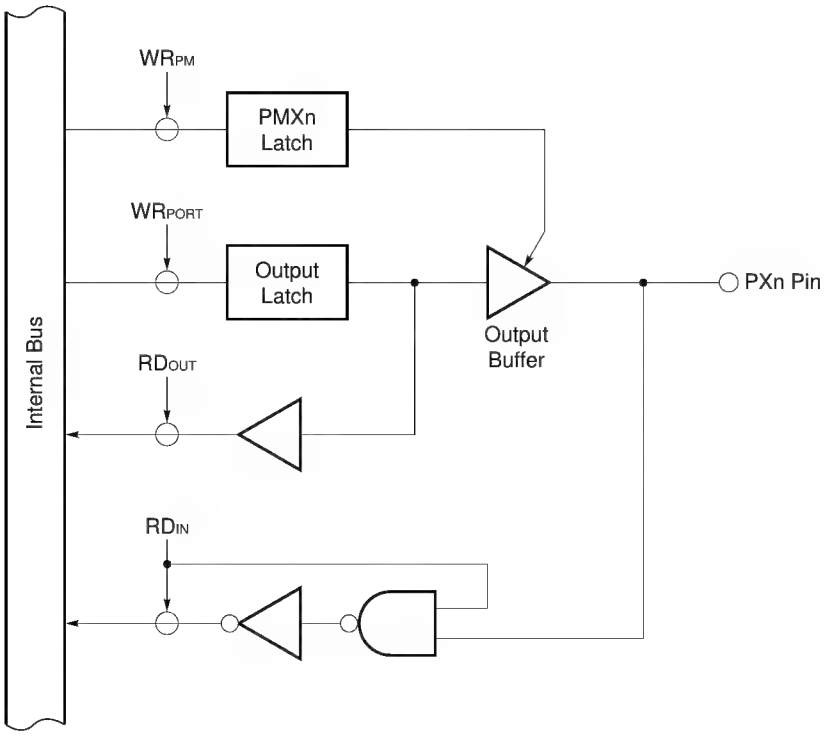
V55PI ports 0, 2-5, 7 and 8 basically consist of the 3-state bi-directional ports shown in Figure 14-1.

The port mode register (PMn) of each port can specify input/output port for each bit.

$\overline{\text{RESET}}$ input resets (1) each bit (PMXn) of the port mode register and makes each bit an input port. At this time, all the port pins enter the high-impedance state. $\overline{\text{RESET}}$ input has no affect on the contents of the output latch.

Hereafter, the shaded areas in tables indicate the states after reset.

Figure 14-1. Ports 0, 2 to 5, 7, 8 Configuration

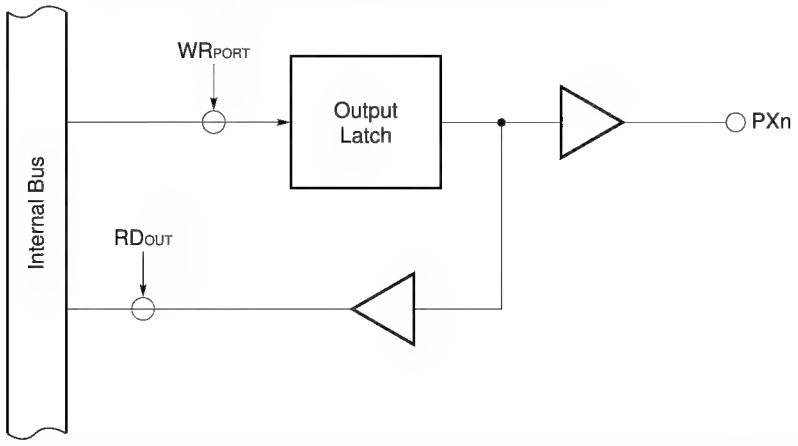


Remark X indicates the port number and n indicates the bit number on the port mode register.

14.2.1 When specified output port (PMXn = 0)

The output latch is enabled and data can be transferred to the output latch by transfer instruction, etc. The contents of the output latch can also be freely set by logic operation instruction. Once written to the output latch, the data is held until the next port operation instruction is executed.

Figure 14-2. Output Port Specification Port



Remark X indicates the port number and n indicates the bit number in the port mode register.

When an output port is read, whether the value of the output latch is read or the pin level is read can be specified by software as a function which checks the output state of the output port.

A port read control register (PRDC) is provided as a control register for this purpose.

	7	6	5	4	3	2	1	0
PRDC	0	0	0	0	0	0	0	PRDC0

Port	Operation	PRDC0	Data
Input port	Read	0	Pin level
		1	Read 0
Output port	Read	0	Output latch value
		1	Pin level

An output port is operated as follows by CPU SFR access:

Read operation : <1> PRDC = 0 ... Reads the output latch value.
<2> PRDC = 1 ... Reads the pin level.

Write operation : Writes data to the port output latch and simultaneously outputs it from the output pin.

When performing pin check by user program by PRDC register operation, pay careful attention to the following points.

(1) Port bit operation instruction at PRDC = 1

In the pin access mode (PRDC = 1), port bit operation instructions do not operate normally. At the end of port check, always return to the normal mode (PRDC = 0).

(2) Interrupt generation at PRDC = 1

When an interrupt is generated in the pin access mode, port operation (bit setting, write to port, etc.) may be performed in the interrupt handling routine and may cause erroneous operation. The interrupt disable state (DI state) must always be set before the check operation is started. Also, do not use port operation macro service.

(3) Nonmaskable interrupt generation

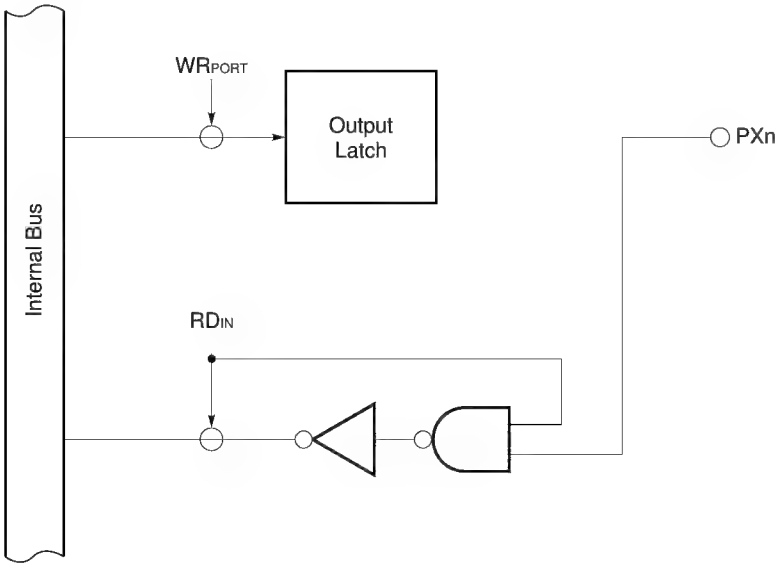
Since the generation of nonmaskable interrupts cannot be avoided, take one of the following measures in the program, depending on the system.

- Do not perform port operation with a nonmaskable interrupt routine.
- Save the PRDC bit at the beginning of a nonmaskable interrupt handling routine and restore the PRDC bit when returning to the main routine.

14.2.2 When specified input port (PMXn = 1)

The port pin level can be loaded by transfer instruction, etc. Writing to data latch is possible in this case also. All data is stored to the output latch regardless of the port input/output specification. However, since the output buffer of a bit specified an input port enters the high-impedance state, the data is not output from the port pin. (When a pin specified an input port is switched to an output port, the contents of the output latch are output from at the port pin.) The contents of the output latch specified as an input port cannot be read.

Figure 14-3. Input Port Specification Port



Remark X indicates the port number and n indicates the bit number in the port mode register.

An input port is operated as follows by CPU SFR access.

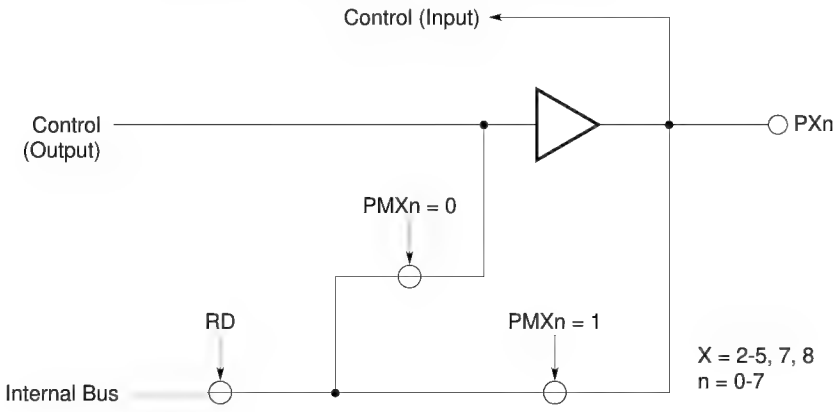
Read operation : <1> PRDC = 0 ..... Reads the pin level.
<2> PRDC = 1 ..... Reads “0”.

Write operation : <1> Input only port ..... Port is not affected.
The CPU executes the SFR access cycle.
<2> Input/output port ... Writing to port output latch is performed, but the data is not output from the pin. (The written data is held and is output from the pin when the pin is made an output port.)

14.2.3 Control specification (PMXn = 1)

Ports can be used as control signal input or output pins in bit or byte units without regard to the port mode register (PMX) setting by setting (1) a port mode control register (PMCX) bit. When a pin is used as a control signal, the status of the control signal can be read by executing a port access instruction.

Figure 14-4. Control Specification Port



Remark X indicates the port number and n indicates the bit number in the port mode register.

(1) When port is control signal output

If a port read instruction is executed when each bit (PMXn) of the port mode register is set (1), the control signal pin status can be read.

If a port read instruction is executed when the port mode register is reset (0), the internal control signal status can be read.

(2) When port is control signal input

When a port read instruction is executed, the control signal pin status can be read only when each bit (PMXn) of the port mode register is set (1).

When a port read instruction is executed when the port mode register is reset (0), the output latch status is read.

14.3 Port Pins Functions

A corresponding control register is provided for each port. Whether each bit of the port is used as a general-purpose port or control signal input/output pin or input or output in the port mode can be specified by control register bit.

14.3.1 Port 0 (P00 to P07)

This is an 8-bit general-purpose input/output port. Input/output can be switched in bit units by port 0 mode register specification.

Table 14-1. Port 0 Operation (n = 0 to 7)

	Port Mode	
	PM0n = 1	PM0n = 0
P00	Input port	Output port
P01	Input port	Output port
P02	Input port	Output port
P03	Input port	Output port
P04	Input port	Output port
P05	Input port	Output port
P06	Input port	Output port
P07	Input port	Output port

(1) Port 0 mode register (PM0)

The PM0 register is an 8-bit register that specifies port 0 input/output in bit units. It can be read/written by 8-bit or 1-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

PM0

7	6	5	4	3	2	1	0
PM0 7	PM0 6	PM0 5	PM0 4	PM0 3	PM0 2	PM0 1	PM0 0

PM0n	Input/Output Specification
0	Output
1	Input

14.3.2 Port 1 (P10 to P16)

This is a 7-bit special input port. P10-P16 function as external interrupt input, timer unit capture trigger input, or A/D converter trigger input pins, as well as input only ports.

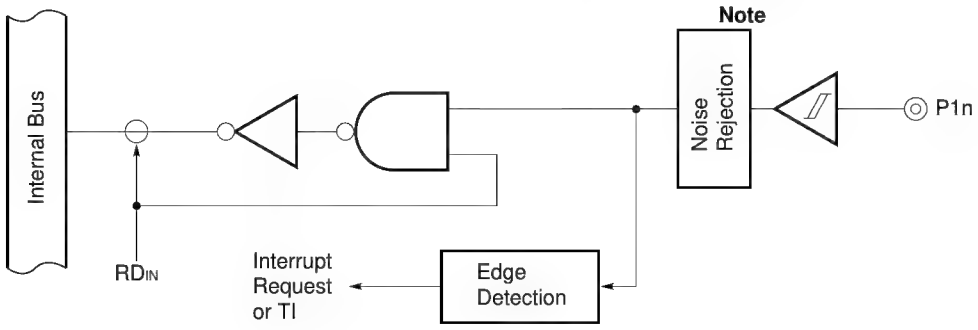
This port also function as an interrupt input pin, but the interrupt function and input port function cannot be switched.

When desiring to use the input port function only, mask the pertinent interrupt and capture functions. (Interrupts, other than NMI, are masked and the capture operation is disabled automatically by system reset.)

Table 14-2. Port 1 Operation

	Function		
P10	NMI input port	or	input only port
P11	INTP0 input port	or	input only port
P12	INTP1 input port	or	input only port
P13	INTP2 input port	or	input only port
P14	INTP3 input/TI input port	or	input only port
P15	INTP4 input port	or	input only port
P16	INTP5 input port	or	input only port

Figure 14-5. Port 1 Block Diagram



Note P10/NMI pin only.

Remark n = 0 to 6

14.3.3 Port 2 (P20 to P25)

This is a 6-bit general-purpose input/output port.

Input/output can be specified in bit units by port 2 mode register (PM2).

This port can also be used as PWM output and timer unit output by port 2 control register (PMC2) setting.

Table 14-3. Port 2 Operation (n = 0 to 5)

	Control Mode	Port Mode	
	PMC2n = 1	PMC2n = 0	
		PM2n = 1	PM2n = 0
P20	PWM output	Input port	Output port
P21	TO00 output	Input port	Output port
P22	TO01 output	Input port	Output port
P23	TO20 output	Input port	Output port
P24	TO21 output	Input port	Output port
P25	TO30 output	Input port	Output port

(1) Port 2 mode control register (PMC2)

The PMC2 register is an 8-bit register that sets the port 2 port/control mode in bit units.

It can be read/written by 8-bit or 1-bit operation memory access.

When the corresponding bit of the PMC2 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the bits of the PMC2 register are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC2	0	0	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20

PMC2n	Mode Specification
0	Port mode
1	Control mode

(2) Port 2 mode register (PM2)

The PM2 register is an 8-bit register that specifies port 2 input/output in bit units.

When the corresponding bit of the PMC2 register is reset (0) (port mode), that bit of the PM2 register is enabled.

This register can be read/written by 8-bit or 1-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

	7	6	5	4	3	2	1	0
PM2	1	1	PM25	PM24	PM23	PM22	PM21	PM20

PM2n	Input/Output Specification
0	Output
1	Input

14.3.4 Port 3 (P30 to P36)

This is a 7-bit general-purpose input/output port. Input/output can be specified in bit units by port 3 mode register (PM3) setting.

This port can also be used as serial input/output by port 3 mode control register (PMC3) setting.

Table 14-4. Port 3 Operation (n = 0 to 6)

	Control Mode	Port Mode	
	PMC3n = 1	PMC3n = 0	
		PM3n = 1	PM3n = 0
P30	TxD0 output/SB0 input/output/SO0 output	Input port	Output port
P31	RxD0 input/SB1 input/output/SI0 input	Input port	Output port
P32	TxC output/$\overline{\text{SCK0}}$ input/output	Input port	Output port
P33	$\overline{\text{CTS0}}$ input	Input port	Output port
P34	TxD1 output/SO1 output	Input port	Output port
P35	RxD1 input/SI1 input	Input port	Output port
P36	$\overline{\text{CTS1}}$ input/$\overline{\text{SCK1}}$ input/output	Input port	Output port

(1) Port 3 mode control register (PMC3)

The PMC3 register is an 8-bit register that sets port 3 port/control mode in bit units.

It can read/written by 8-bit or 1-bit operation memory access.

When the corresponding bit of the PMC3 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the bits of the PMC3 register are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC3	0	PMC3 6	PMC3 5	PMC3 4	PMC3 3	PMC3 2	PMC3 1	PMC3 0

PMC3n	Mode Specification
0	Port mode
1	Control mode

(2) Port 3 mode register (PM3)

The PM3 register is an 8-bit register that specifies port 3 input/output in bit units.

When the corresponding bit of the PMC3 register is reset (0) (port mode), that bit of the PM3 register is enabled.

This register can be read/written by 8-bit or 1-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

	7	6	5	4	3	2	1	0
PM3	1	PM3 6	PM3 5	PM3 4	PM3 3	PM3 2	PM3 1	PM3 0

PM3n	Input/Output Specification
0	Output
1	Input

14.3.5 Port 4 (P40 to P47)

This is an 8-bit general-purpose input/output port. Input/output can be specified in bit units by port 4 mode register (PM4) setting.

This port can also be used as parallel interface data input/output (PD0 to PD7) by port 4 mode control register (PMC4) setting.

Table 14-5. Port 4 Operation (n = 0 to 7)

	Control Mode	Port Mode	
	$PMC4_7$ = 1	$PMC4_7$ = 0	
		PM4n = 1	PM4n = 0
P40	PD0 input/output	Input port	Output port
P41	PD1 input/output	Input port	Output port
P42	PD2 input/output	Input port	Output port
P43	PD3 input/output	Input port	Output port
P44	PD4 input/output	Input port	Output port
P45	PD5 input/output	Input port	Output port
P46	PD6 input/output	Input port	Output port
P47	PD7 input/output	Input port	Output port

(1) Port 4 mode control register (PMC4)

The PMC4 register is an 8-bit register that sets port 4 port/control mode in byte units.

It can be read/written by 8-bit or 1-bit operation memory access.

When the corresponding bit of the PMC4 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the PMC4 bits are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC4	$PMC4_7$	0	0	0	0	0	0	0

$PMC4_7$	Mode Specification
0	Port mode
1	Control mode

(2) Port 4 mode register (PM4)

The PM4 register is an 8-bit register that specifies port 4 input/output in bit units.

When the corresponding bit of the PMC4 register is reset (0) (in port mode), that bit of the PM4 register is enabled. This register can be read/written by 8-bit or 1-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

	7	6	5	4	3	2	1	0
PM4	PM4 7	PM4 6	PM4 5	PM4 4	PM4 3	PM4 2	PM4 1	PM4 0

PM4n	Input/Output Specification
0	Output
1	Input

14.3.6 Port 5 (P50 to P52)

This is a 3-bit general-purpose input/output port. Input/output can be specified in bit units by port 5 mode register (PM5).

This ports can also be used as parallel interface control signal by port 5 mode control register (PMC5) setting.

Table 14-6. Port 5 Operation (n = 0 to 2)

	Control Mode	Port Mode	
	PMC5n = 1	PMC5n = 0	
		PM5n = 1	PM5n = 0
P50	$\overline{\text{DATASTB}}$ input/output	Input port	Output port
P51	$\overline{\text{ACK}}$ input/output	Input port	Output port
P52	BUSY input/output	Input port	Output port

(1) Port 5 mode control register (PMC5)

The PMC5 register is an 8-bit register that sets port 5 port/control mode in bit units.

It can be read/written by 8-bit or 1-bit operation memory access.

When the corresponding bit of the PMC5 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the PMC5 register bits are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC5	0	0	0	0	0	PMC5 2	PMC5 1	PMC5 0

PMC5n	Mode Specification
0	Port mode
1	Control mode

(2) Port 5 mode register (PM5)

The PM5 register is an 8-bit register that specifies the port 5 input/output in bit units.

When the corresponding bit of the PMC5 register is reset (0) (port mode), that bit of the PM5 register is enabled.

This register can be read/written by 8/1 operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

	7	6	5	4	3	2	1	0
PM5	1	1	1	1	1	PM5 2	PM5 1	PM5 0

PM5n	Input/Output Specification
0	Output
1	Input

14.3.7 Port 6 (P60 to P63)

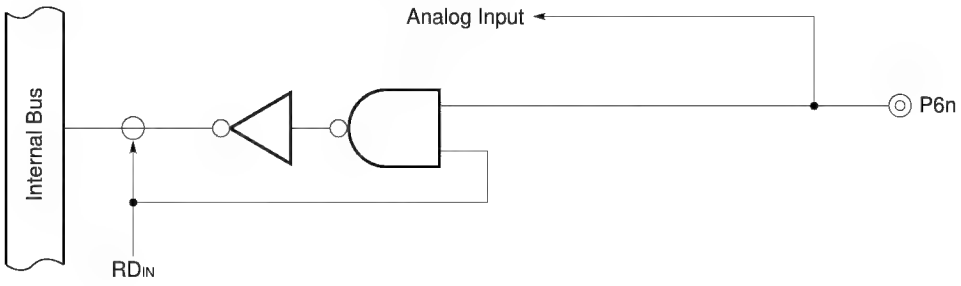
This is a 4-bit general-purpose input port. It also functions as an A/D converter analog input.

This port is also used as an analog input pin, but it cannot be switched between input port and analog input pin.

Table 14-7. Port 6 Operation

	Function
P60	ANI0 input port or input only port
P61	ANI1 input port or input only port
P62	ANI2 input port or input only port
P63	ANI3 input port or input only port

Figure 14-6. Port 6 Block Diagram



Remark n = 0 to 3

14.3.8 Port 7 (P70 to P77)

This is an 8-bit general-purpose input/output port. Input/output specification can be specified in bit units by port 7 mode register (PM7) setting.

This port can also be used as a real-time output port by port 7 mode control register (PMC7) setting. For a detailed description of the real-time output port, see section **CHAPTER 10 REAL-TIME OUTPUT PORT FUNCTION**.

Table 14-8. Port 7 Operation (n = 0 to 7)

	Control Mode	Port Mode	
	PMC7n = 1	PMC7n = 0	
		PM7n = 1	PM7n = 0
P70	RTP0 output	Input port	Output port
P71	RTP1 output	Input port	Output port
P72	RTP2 output	Input port	Output port
P73	RTP3 output	Input port	Output port
P74	RTP4 output	Input port	Output port
P75	RTP5 output	Input port	Output port
P76	RTP6 output	Input port	Output port
P77	RTP7 output	Input port	Output port

(1) Port 7 mode control register (PMC7)

The PMC7 register is an 8-bit register that sets port 7 port/control mode in units.

It can be read/written by 8/1 bit operation memory access.

When the corresponding bit of the PMC7 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the PMC7 register bits are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC7	PMC7 7	PMC7 6	PMC7 5	PMC7 4	PMC7 3	PMC7 2	PMC7 1	PMC7 0

PMC7n	Mode Specification
0	Port mode
1	Control mode

(2) Port 7 mode register (PM7)

The PM7 register is an 8-bit register that specifies port 7 input/output in bit units.

When the corresponding bit of the PMC7 register is reset (0) (port mode), that bit of the PM7 register is enabled.

This register can be read/written by 8-bit or 1-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and make them input ports.

	7	6	5	4	3	2	1	0
PM7	PM7 7	PM7 6	PM7 5	PM7 4	PM7 3	PM7 2	PM7 1	PM7 0

PM7n	Input/Output Specification
0	Output
1	Input

14.3.9 Port 8 (P80, P81)

This is a 2-bit general-purpose input/output port. Input/output can be specified in bit units by port 8 mode register (PM8) setting.

This port is also used as a DMA control pin by port 8 mode control register (PMC8) setting.

Table 14-9. Port 8 Operation (n = 0, 1)

	Control Mode	Port Mode	
	PMC8n = 1	PMC8n = 0	
		PM8n = 1	PM8n = 0
P80	DMARQ0 input	Input port	Output port
P81	DMARQ1 input	Input port	Output port

(1) Port 8 mode control register (PMC8)

The PMC8 register is an 8-bit register that sets port 8 port/control mode in bit units.

It can be read/written by 8-bit or 1-bit operation memory access.

When the corresponding bit of the PMC8 register is set (1), that bit enters the control signal input/output mode and when the bit is reset (0), it enters the port mode. At $\overline{\text{RESET}}$ input, all the PMC8 register bits are reset (0) and enter the port mode.

	7	6	5	4	3	2	1	0
PMC8	0	0	0	0	0	0	PMC81	PMC80

PMC8n	Mode Specification
0	Port mode
1	Control mode

(2) Port 8 mode register (PM8)

The PM8 register is an 8-bit register that specifies port 8 input/output in bit units.

When the corresponding bit of the PMC8 register is reset (0) (port mode), that bit of the PM8 register is enabled.

This register can be read/written by 8-bit operation memory access.

$\overline{\text{RESET}}$ input sets (1) all the bits and makes them input ports.

	7	6	5	4	3	2	1	0
PM8	1	1	1	1	1	1	PM81	PM80

PM8n	Input/Output Specification
0	Output
1	Input

[MEMO]

CHAPTER 15 STANDBY FUNCTION

The V55PI has two modes which control the operation clock as a standby function to reduce the power consumption. The V55PI can be switched to either standby state by dedicated instruction.

15.1 Features

- Recovery from stop state judgment possible
- HALT mode (CPU operation stopped)
- STOP mode (clock oscillation stopped)

15.2 Standby Modes

There are two standby modes: HALT mode and STOP mode. The operation status of each mode is shown in Table 15-1.

Table 15-1. HALT/STOP Mode Operation States

Item		HALT Mode	STOP Mode
Clock oscillator		Operated	Stopped
Internal system clock		Stopped	
16-bit timer		Operated	
Watchdog timer			
HOLD circuit			
Serial interface			
Parallel interface			
A/D converter			
Interrupt request controller			
DMA controller			
★ Port		Input: Hi-z, output: held	Input: Hi-z, output: held
★ $\overline{IORD}$, $\overline{IOWR}$		High level	High level
Bus line	AD0-AD15	Changes according to the DMAC operation status	Held
	A16-A23		
R/W output		High level	High level
Refresh operation		Operated	Stopped
Data hold		CPU status, RAM contents and other internal data all held	CPU status, ROM contents, and other internal data all held
Reset method		• NMI • INTWDT • Maskable interrupt request • $\overline{RESET}$ input	• NMI • $\overline{RESET}$ input

15.3 Control Registers

15.3.1 Standby control register (STBC)

The STBC register is used in recovery from STOP mode judgment. The SBF flag can be set (1) by instruction only. When the power is turned on, the SBF flag is reset (0) only when the voltage rose from 0 V. For the RESET signal and software, the SBF flag is not reset (0), but remains in its previous state.

Therefore, if the SBF flag is set (1) before the system enters the STOP mode, the system initialization routine can judge between starting by power-on and recovery from the STOP mode.

	7	6	5	4	3	2	1	0
STBC	0	0	0	0	0	0	0	SBF

SBF	Operation
0	At power on V_{DD} = 0 [V]
1	Write by instruction

15.4 HALT Mode

15.4.1 Halt mode setting and operation

This mode stops the CPU operation clock.

The power consumption of the entire system can be reduced by setting it to the HALT mode when the CPU is idle. The HALT mode is set by executing a HALT instruction.

In the HALT mode, the CPU clock and program execution are stopped and the previous contents of all the registers and on-chip RAM are held. The status of each hardware is shown in Table 15-1.

When the HALT instruction was executed during DMA transfer, transition to the HALT mode is held up until the end of the transfer bus cycle corresponding to one DMA request.

15.4.2 HALT mode release

(1) Release by interrupt request

The HALT mode is released by nonmaskable interrupt (NMI, INTWDT) request and unmasked maskable interrupt request regardless of their priority.

After HALT mode release by nonmaskable interrupt request, the program always branches to the vector address.

When the system enters the HALT mode in an interrupt handling routine, operation after release differs with the priority of the interrupt being handled and the HALT mode reset source interrupt.

(a) When priority is "release source interrupt > processing interrupt"

After the HALT mode is released, the program branches to vectored interrupt servicing, or switches to the corresponding register bank. (However, for EI state.)

(b) For other priorities or DI state

After the HALT mode is released, the program is executed from the instruction after the HALT instruction.

(c) Release by macro service

Halt mode release after the end of macro service is shown in Table 15-2.

Table 15-2. HALT Mode Release by Macro Service Request

Interrupt Generation Condition after End of Macro Service	HALT Mode Release	Operation after HALT Mode Release	
		EI State	DI State
When established	Release	Branches to interrupt handling routine or executes next instruction.	Executes next instruction.
When not established	Returns to HALT mode after end of macro service.	—	

(2) Release by reset

The HALT mode is released by $\overline{\text{RESET}}$ input. The normal reset operation is performed.

(3) Release by DMA request

When a DMA request is generated in the HALT mode, DMA transfer is executed after the HALT mode is released internally. After DMA transfer execution, the system returns to the HALT mode.

At this time, the contents of the PC are not changed. Therefore, it appears as if the HALT mode was released.

15.5 STOP Mode

15.5.1 STOP mode setting and operation

This mode stops clock oscillation.

It is effective when the entire application system is stopped and realizes ultra-low power consumption. The STOP mode is set by executing a STOP instruction. In the STOP mode, all the clocks stop. Program execution stops, but the contents of all the registers and on-chip RAM before that are preserved. The hardware enters the state shown in **Table 15-1**.

When a STOP instruction was executed during DMA transfer, transition to the STOP mode is held up until the end of the transfer bus cycle corresponding to one DMA request. When execution of a refresh cycle and STOP instruction compete, the STOP instruction is held up until the end of the refresh cycle.

Caution When executing the STOP instruction, be sure to set the CS bit of the ADM register to 0 and stop the conversion operations of the A/D converter.

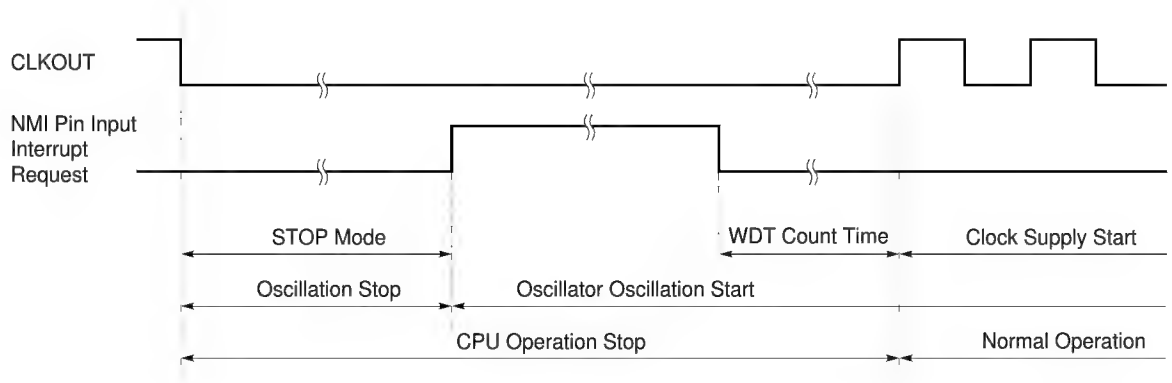
15.5.2 STOP mode release

(1) Reset by NMI request

The internal system clock supply is started after a wait of "high level or low level width after NMI edge detection + watchdog timer overflow time[Note]" as the oscillation stabilization time.

After STOP mode released by nonmaskable interrupt, the program always branches to the vector address.

Figure 15-1. STOP Mode Release by NMI Pin Input



Note The watchdog timer overflow time at STOP mode release is 10.4 ms (when system clock = 12.5 MHz), and 8.1 ms (when system clock = 16 MHz).

At this time, a watchdog timer interrupt (INTWDT) is not generated.

(2) Release by reset

The STOP mode is released by $\overline{\text{RESET}}$ input. When a low level signal is input to the $\overline{\text{RESET}}$ pin, clock oscillation is restarted. The low level must be held until oscillation stabilizes.

CHAPTER 16 CLOCK GENERATOR

The clock generator generates various clock to the CPU and peripheral hardware and controls operation timings.

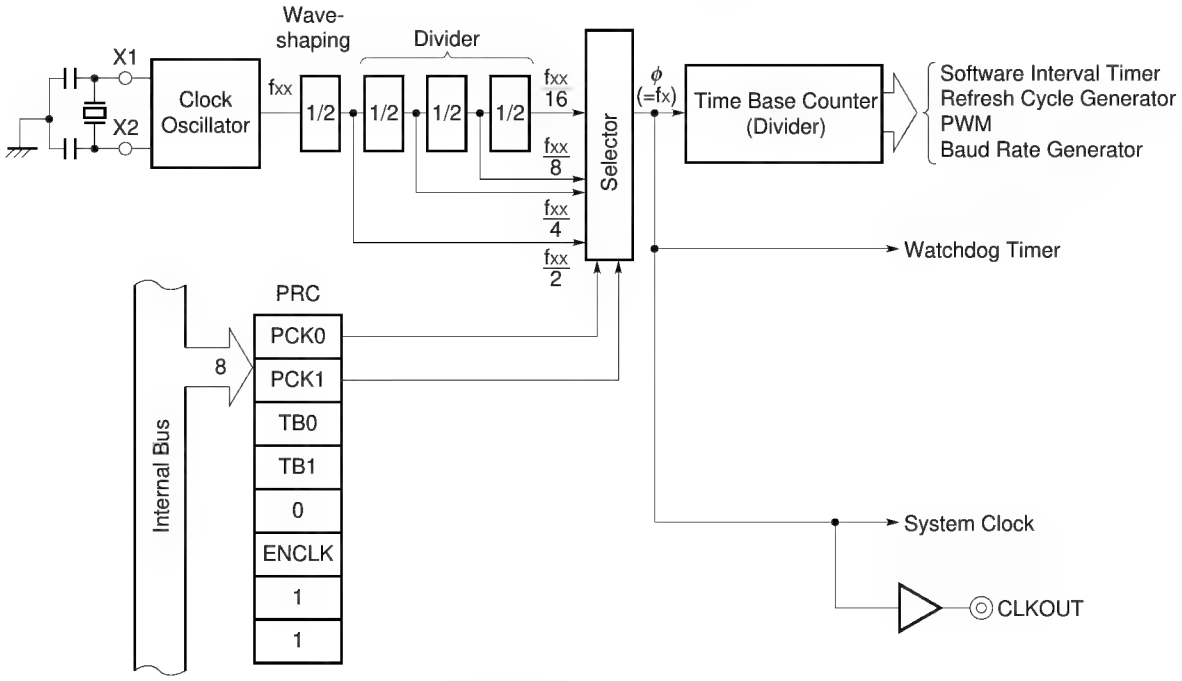
16.1 Features

- On-chip clock generation circuit
- External clock input (X1 input)
- Crystal/ceramic oscillaiton (X1 and X2 input)

16.2 Configuration

The clock generator block diagram is shown in Figure 16-1.

Figure 16-1. Clock Generator



fxx : Oscillation Frequency
φ : System Clock
PRC : Processor Control Register

16.3 Operation

16.3.1 Clock oscillation circuit

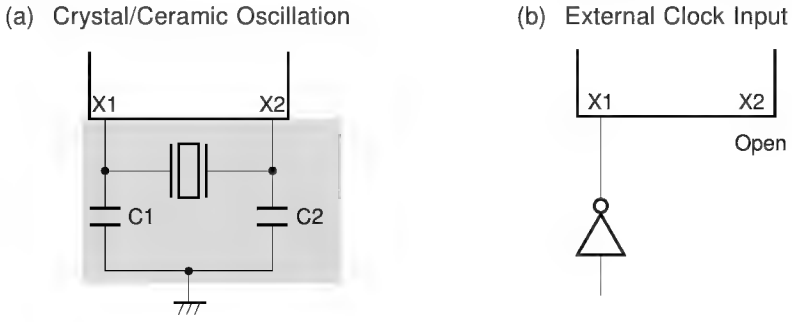
In a clock oscillation circuit, a reference clock is generated by a crystal or ceramic resonator connected to the X1 and X2 pins. The clock oscillation circuit output is waveshaped (1/2 division) and the division ratio is selected by the processor control register (PRC) and used as system clock ϕ.

The system clock ϕ division ratio can be selected from among 1/2, 1/4, 1/8 and 1/16 of the oscillation frequency (fxx) by PRC register PCK1 and PCK0 bit specification.

By slowing down the system clock ϕ, long-term operation with a battery drive system is possible even if the voltage drops.

An external clock can also be input. In this case, input the clock at the X1 pin and open the X2 pin.

Figure 16-2. Clock Generator External Circuits



Cautions 1. Place the oscillator as close as possible to the X1 and X2 pins.

2. Do not run other signal lines within the shaded area.

16.3.2 Time base counter (TBC)

The V55PI shares the divider (time base counter TBC) that divided the internal system clock ϕ with the timer units.

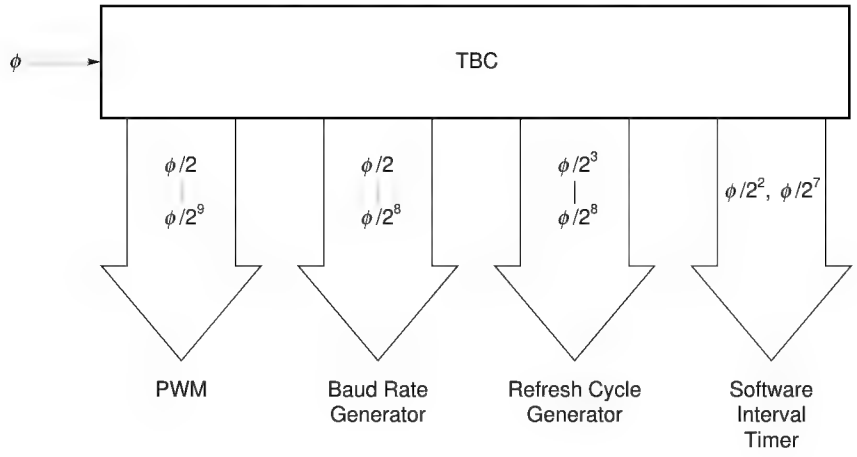
The TBC cannot be read/written by instruction.

The TBC tap outputs (2^n divided clock) are supplied to the count clock to the following units:

(1) Refresh cycle generator
(2) Software interval timer
(3) PWM unit
(4) Baud rate generator

The TBC is cleared to zero only by reset input and is constantly incremented thereafter. The TBC is stopped in the STOP mode. The TBC block diagram is shown in Figure 16-3.

Figure 16-3. Time Base Counter (TBC) Configuration



16.4 Processor Control Register (PRC)

The PRC register is an 8-bit register that specifies the CPU operation clock and software interval timer input clock and centrally controls the items related to CPU and internal system control.

After reset, the PRC register value becomes EEH.

Always set bit 4 to "0" and bits 6 and 7 to "1".

	7	6	5	4	3	2	1	0
PRC	1	1	ENCLK	0	TB1	TB0	PCK1	PCK0

System Clock ϕ Division Ratio Specification

PCK1	PCK0	Specification
0	0	ϕ = oscillation frequency (f_{XX}) × (1/2)
0	1	ϕ = oscillation frequency (f_{XX}) × (1/4)
1	0	ϕ = oscillation frequency (f_{XX}) × (1/8)
1	1	ϕ = oscillation frequency (f_{XX}) × (1/16)

Software Interval Timer Input Clock Specification

TB1	TB0	Input Clock Frequency	Maximum Interval Time	
			ϕ = 12.5 MHz	ϕ = 16 MHz
0	0	$\phi/2^2$	21.0 ms	16.4 ms
0	1	$\phi/2^7$	671 ms	524.2 ms
1	0	Stop operation	∞	∞
1	1			

CLKOUT Pin System Clock Output Control Specification

ENCLK	Specification
0	Disable output (stopped by low level)
1	Enable output

(CLKOUT pin output level in standby mode)

HALT mode: Output continued
STOP mode: Output stopped
(previous level held)

Remark The shaded area indicates status after a reset.

PCK1 PCK0 System clock division ratio specification

After the oscillator output is divided by the value specified by this bit, it is used as the system clock ϕ.

When the external crystal oscillation frequency fx = 32.0 MHz, immediately after reset, the system clock ϕ becomes the following value:

ϕ = (fx) × 1/8 = 4.0 MHz

To fully display the performance of the V55PI, set PCK1=PCK0=0 and operate at ϕ = 16 MHz.

Caution Rewriting this bit changes the frequency of all clocks generated by the system clock.

TB1 TB0 Software interval timer input count clock frequency specification and operation control.

Software interval timer stop/start can be controlled with the TB1 bit. When counter operation is restarted after setting the bit from 1 to 0, the software timer counter (STC) is cleared to 0 and incremented.

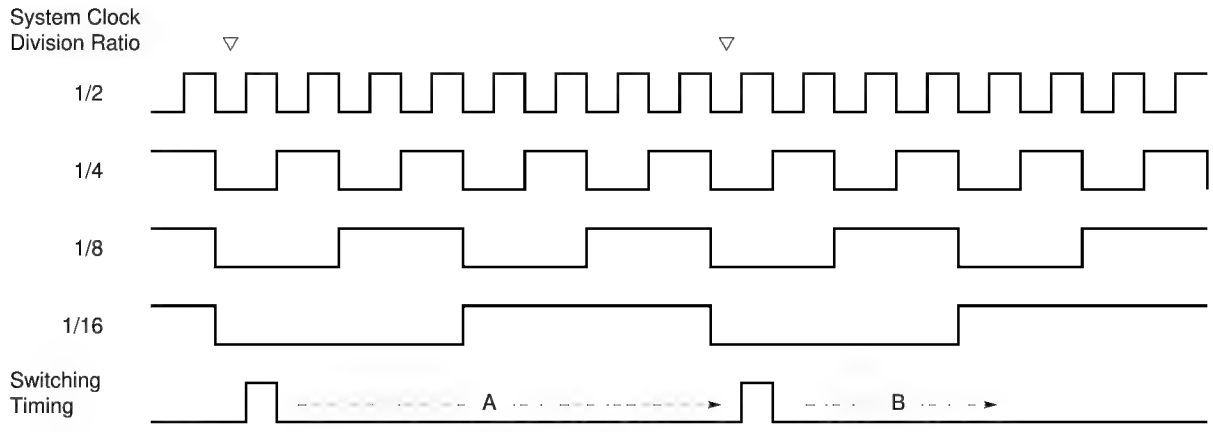
16.5 Clock Switching Timing

System clock switching is not performed immediately after PRC register rewriting, but during a fixed machine cycle by the clock preceding switching.

Therefore, when a standby instruction is executed after system clock switching and when the system clock must be used as the timer, baud rate generator, refresh circuit, etc. operation clock after switching, a standby instruction must be executed or the pertinent circuit must be enabled after this switching time.

An switching timing example is shown below.

Figure 16-4. System Clock Switching Timing



The system clock frequency is switched from the clock after the 1/2, 1/4, 1/8 and 1/16 clocks all became low level.

Therefore, when a PRC register PCK1 and PCK0 bit write cycle (internal write cycle) is executed by MOV instruction, etc. within period A (between ▽ - ▽), the system is operated by the new clock beginning from the timing of B.

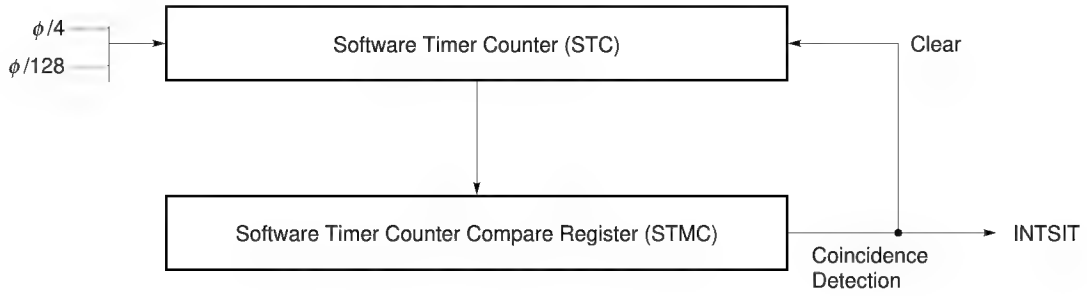
CHAPTER 17 SOFTWARE INTERVAL TIMER FUNCTION

The V55PI has a on-chip 16-bit software interval timer as a software timer function and clock function timer.

17.1 Configuration

The software interval timer block diagram is shown in Figure 17-1.

Figure 17-1. Software Interval Timer Configuration



17.2 Control Registers

(1) Processor control register (PRC)

The input clock (count clock) to the software interval timer can be selected with the processor control register (PRC) TB0 bit. Timer start/stop can be controlled with the PRC register TB1 bit.

	7	6	5	4	3	2	1	0
PRC	1	1	ENCLK	0	TB1	TB0	PCK1	PCK0

Software Interval Timer Input Clock Specification

TB1	TB0	Input Clock Frequency	Maximum Interval Time	
			ϕ = 12.5 MHz	ϕ = 16 MHz
0	0	$\phi/2^2$	21.0 ms	16.4 ms
0	1	$\phi/2^7$	671 ms	524 ms
1	0	Stop operation	∞	∞
1	1			

Remark The shaded area indicates status after a reset.

(2) Software timer counter (STC)

The STC is a 16-bit counter that is counted up by the timing of the input clock specified by the PRC register. Its value is compared to the value of the software timer counter compare register and when coincidence is detected, an interrupt request is generated. After coincidence detection, the contents of the STC are cleared (0).

The STC can only be read by 16-bit operation instruction.

(3) Software timer counter compare register (STMC)

The interval time of the interrupt request that is generated from the software interval timer is specified by the set value of the 16-bit software timer counter compare register (STMC).

The value after a reset of the STMC register become FFFFH.

The STMC register can be read/written by 16-bit operation instruction.

17.3 Interrupt Request

The software timer counter (STC) is counted up by the input clock specified by the PRC register and generates an interval interrupt (INTSIT) when its value coincides with the software timer counter compare register (STMC) set value.

The interval interrupt generation period is:

Generation period = STMC register set value/input clock frequency

When "0" is set in the STMC register, an interrupt is generated at every two clocks without regard to the input clock selection contents.

The time from immediately after setting to generation of the first interrupt is undefined.

INTSIT interrupt can be masked by interrupt request control register IC32.

CHAPTER 18 CODEC FUNCTIONS

The V55PI has nine types of codec instructions.

Using these instructions, the V55PI can achieve not only MH[Note 1] encoding of binary image data but also MR encoding[Note 2], which used to require dedicated devices such as ACEE (advanced compression expansion engine), through small-scale and high-speed software codec.

Notes 1. MH: Modified Huffman
2. MR: Modified Relative element designate

18.1 Features

The V55PI has the following nine codec instructions (4 compression instructions and 5 expansion instructions).

- Compression instructions
  (1) Turning point table create instruction: COLTRP
  (2) Data transmit instruction (transmission of EOL[Note 1], FILL, RTC[Note 2], etc.): ALBIT
  (3) MH encode instruction: MHENC
  (4) MR encode instruction: MRENC

- Expansion instructions
  (5) EOL detect instruction: SCHEOL
  (6) 1 bit (tag) detect instruction: GETBIT
  (7) MH decode turning point table create instruction: MHDEC
  (8) MR decode turning point table create instruction: MRDEC
  (9) Pixel data create instruction: CNVTRP

MH and MR encoding is carried out as shown in Figure 18-1 and MH and MR decoding is carried out as shown in Figure 18-2 using these instructions.

Notes 1. EOL: End of Line
2. RTC: Return To Control

Caution For compression/expansion using the V55PI codec instructions, make the necessary specification noting the following.
- **Execute compression/expansion for each line.**
- **Task switching and interruption occur during compression.**
- **The number of bits per line to be processed should remain the same during one-page processing.**
- **For the data of more than 64 Kbytes which occupies several segments, the segment value should be changed.**

Figure 18-1. MH, MR Encoding Process Flow

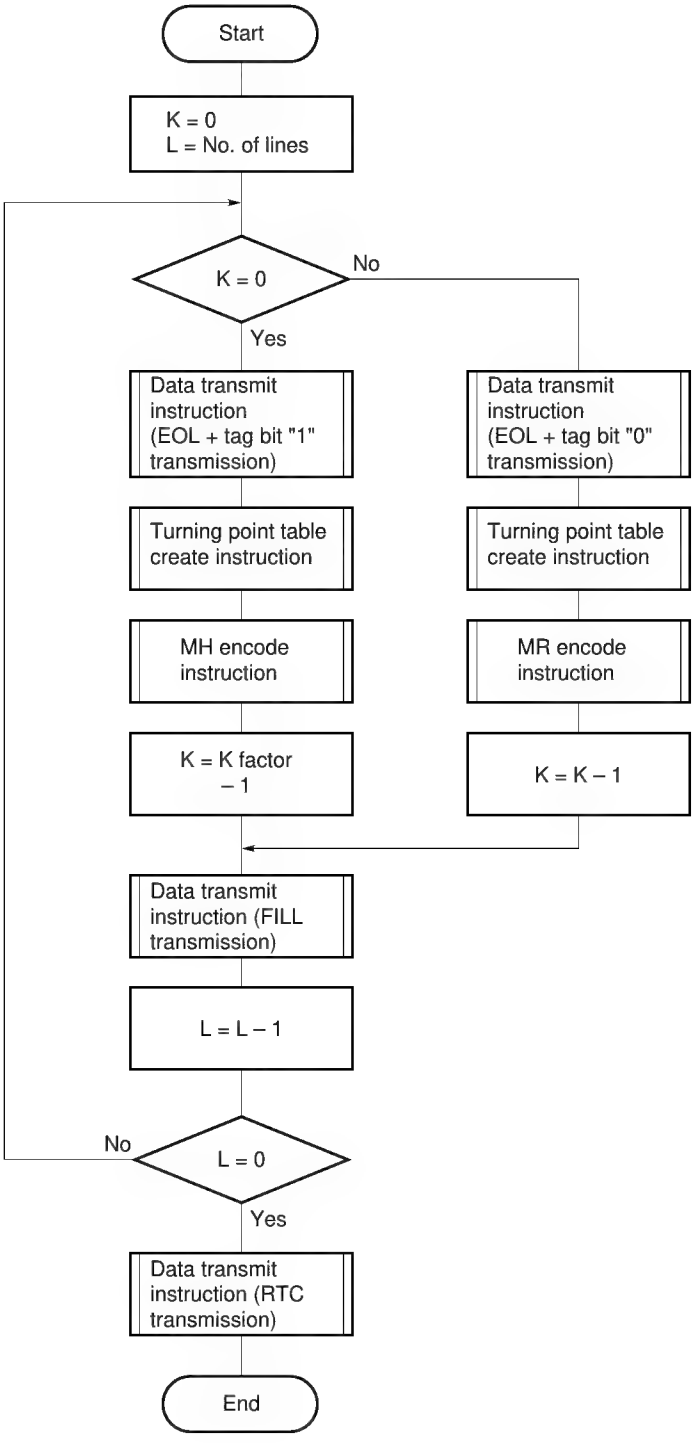
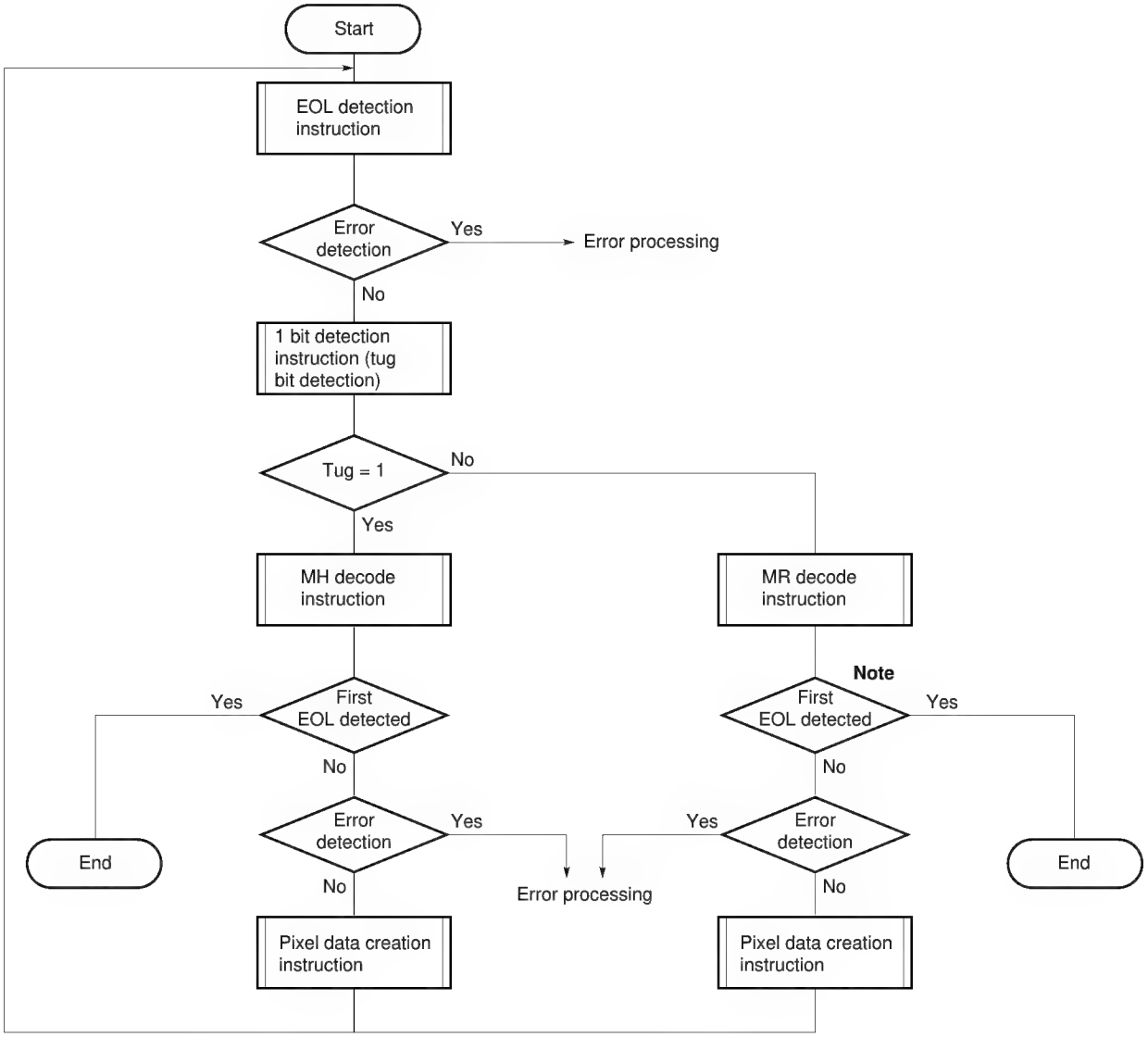


Figure 18-2. MH, MR Decoding Process Flow



Note RTC is detected by two EOLs.

18.2 Encoding/Decoding Methods of Binary Image Data

This section describes the encoding (compression) and decoding (expansion) methods of binary image data.

Based on ITU-T (International Telecommunications Union) recommendations, MH encoding, MR encoding, and MMR[Note] encoding are adopted as the standard encoding (compression) methods of binary image data. Decoding (expansion) is performed in the reverse procedure of encoding.

For details of ITU-T recommendations, refer to "ITU-T White Book G3/G4 Facsimile-Related T Series Recommendation" (Published by New Japan ITU Association).

Note MMR: Modified MR

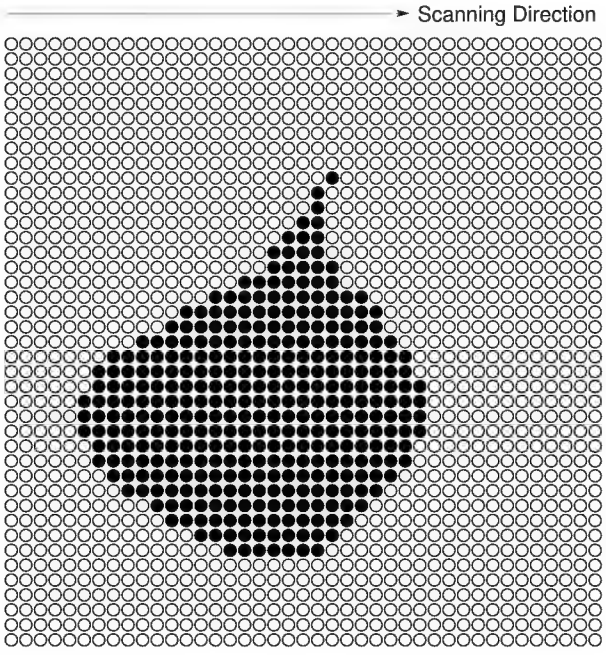
Before each method is explained, the basic concept behind the methods, run-length encoding, is explained.

18.2.1 Run-length encoding

When an image such as the one shown in Figure 18-3 is input as binary image data, it is input into the memory as the bit map data shown in Figure 18-4, which means that white pixels are expressed as 0 and black pixels are expresses as 1.

Figure 18-3. Sample Picture

Figure 18-4. Bit Map Data



Remark ○ : Bit 0
● : Bit 1

When this data is scanned in the arrow direction shown in Figure 18-4, first 0 bits continue, and at a certain point, the bits change to 1 (turning point). As the data is further scanned, the bits change to 0 again. Run-length encoding is the replacing the number of pixels (called run-length) of the same color (white or black) between one turning point and another with codes using a certain method.

For example, if run-length itself is used as the code, the bit map data shown in Figure 18-4 is replaced by the numeric strings shown in Figure 18-5. Although the original data amount is 200 bytes (40 × 40 = 1600 bits = 200 bytes), this will be compressed to several dozen bytes after encoding. By increasing the resolution of the scanner, this difference will further increase.

Figure 18-5. Run-length Encoding

```
341 1 38 1 39 1 38 2 37 3 36 4 36 5 37 8 30 11 27 13 25 15 23 18 20 21 18
22 18 23 16 24 16 24 16 24 17 22 18 22 20 19 23 16 25 14 28 11 31 7 259
```

Decoding the code and playing back the original image can be realized easily using the reverse procedure of encoding as long as the color of the starting point of scanning and the horizontal size of the image is known. Furthermore, the played back image is exactly the same as the original one.

The above has been the concept of run-length encoding. The encoding methods recommended by ITU-T explained hereafter are based on this run-length encoding concept.

18.2.2 MH encoding

The MH encoding is also known as the on-dimension encoding method.

In this encoding method, the run-length is counted for every horizontal line of the image data. The codes corresponding to each run-length shown in Table 18-1 are used. There are two types of codes-make-up code and terminate code. The terminate code is assigned if the run-length is less than 64. when the run-length is more than 64, the make-up code and terminate code are assigned together (See Examples 1 and 2).

Example 1. When the run-length is 231

$$231 = 64 \times 3 + 39 = \underline{192} + \underline{39}$$

Terminate Code

Make-up Code

2. When the run-length is 128

$$128 = 64 \times 2 + 0 = \underline{128} + \underline{0}$$

Terminate Code

Make-up Code

Table 18-1. Run-length Code

(a) Terminate Code

Run-Length	White Run Code Characters	Black Run Code Characters	Run-Length	White Run Code Characters	Black Run Code Characters
0	0 0 1 1 0 1 0 1	0 0 0 0 0 1 1 0 1 1 1	32	0 0 0 1 1 0 1 1	0 0 0 0 0 1 1 0 1 0 1 0
1	0 0 0 1 1 1	0 1 0	33	0 0 0 1 0 0 1 0	0 0 0 0 0 1 1 0 1 0 1 1
2	0 1 1 1	1 1	34	0 0 0 1 0 0 1 1	0 0 0 0 1 1 0 1 0 0 1 0
3	1 0 0 0	1 0	35	0 0 0 1 0 1 0 0	0 0 0 0 1 1 0 1 0 0 1 1
4	1 0 1 1	0 1 1	36	0 0 0 1 0 1 0 1	0 0 0 0 1 1 0 1 0 1 0 0
5	1 1 0 0	0 0 1 1	37	0 0 0 1 0 1 1 0	0 0 0 0 1 1 0 1 0 1 0 1
6	1 1 1 0	0 0 1 0	38	0 0 0 1 0 1 1 1	0 0 0 0 1 1 0 1 0 1 1 0
7	1 1 1 1	0 0 0 1 1	39	0 0 1 0 1 0 0 0	0 0 0 0 1 1 0 1 0 1 1 1
8	1 0 0 1 1	0 0 0 1 0 1	40	0 0 1 0 1 0 0 1	0 0 0 0 0 1 1 0 1 1 0 0
9	1 0 1 0 0	0 0 0 1 0 0	41	0 0 1 0 1 0 1 0	0 0 0 0 0 1 1 0 1 1 0 1
10	0 0 1 1 1	0 0 0 0 1 0 0	42	0 0 1 0 1 0 1 1	0 0 0 0 1 1 0 1 1 0 1 0
11	0 1 0 0 0	0 0 0 0 1 0 1	43	0 0 1 0 1 1 0 0	0 0 0 0 1 1 0 1 1 0 1 1
12	0 0 1 0 0 0	0 0 0 0 1 1 1	44	0 0 1 0 1 1 0 1	0 0 0 0 0 1 0 1 0 1 0 0
13	0 0 0 0 1 1	0 0 0 0 0 1 0 0	45	0 0 0 0 0 1 0 0	0 0 0 0 0 1 0 1 0 1 0 1
14	1 1 0 1 0 0	0 0 0 0 0 1 1 1	46	0 0 0 0 0 1 0 1	0 0 0 0 0 1 0 1 0 1 1 0
15	1 1 0 1 0 1	0 0 0 0 1 1 0 0 0	47	0 0 0 0 1 0 1 0	0 0 0 0 0 1 0 1 0 1 1 1
16	1 0 1 0 1 0	0 0 0 0 0 1 0 1 1 1	48	0 0 0 0 1 0 1 1	0 0 0 0 0 1 1 0 0 1 0 0
17	1 0 1 0 1 1	0 0 0 0 0 1 1 0 0 0	49	0 1 0 1 0 0 1 0	0 0 0 0 0 1 1 0 0 1 0 1
18	0 1 0 0 1 1 1	0 0 0 0 0 0 1 0 0 0	50	0 1 0 1 0 0 1 1	0 0 0 0 0 1 0 1 0 0 1 0
19	0 0 0 1 1 0 0	0 0 0 0 1 1 0 0 1 1 1	51	0 1 0 1 0 1 0 0	0 0 0 0 0 1 0 1 0 0 1 1
20	0 0 0 1 0 0 0	0 0 0 0 1 1 0 1 0 0 0	52	0 1 0 1 0 1 0 1	0 0 0 0 0 0 1 0 0 1 0 0
21	0 0 1 0 1 1 1	0 0 0 0 1 1 0 1 1 0 0	53	0 0 1 0 0 1 0 0	0 0 0 0 0 0 1 1 0 1 1 1
22	0 0 0 0 0 1 1	0 0 0 0 0 1 1 0 1 1 1	54	0 0 1 0 0 1 0 1	0 0 0 0 0 0 1 1 1 0 0 0
23	0 0 0 0 1 0 0	0 0 0 0 0 1 0 1 0 0 0	55	0 1 0 1 1 0 0 0	0 0 0 0 0 0 1 0 0 1 1 1
24	0 1 0 1 0 0 0	0 0 0 0 0 0 1 0 1 1 1	56	0 1 0 1 1 0 0 1	0 0 0 0 0 0 1 0 1 0 0 0
25	0 1 0 1 0 1 1	0 0 0 0 0 0 1 1 0 0 0	57	0 1 0 1 1 0 1 0	0 0 0 0 0 1 0 1 1 0 0 0
26	0 0 1 0 0 1 1	0 0 0 0 1 1 0 0 1 0 1 0	58	0 1 0 1 1 0 1 1	0 0 0 0 0 1 0 1 1 0 0 1
27	0 1 0 0 1 0 0	0 0 0 0 1 1 0 0 1 0 1 1	59	0 1 0 0 1 0 1 0	0 0 0 0 0 0 1 0 1 0 1 1
28	0 0 1 1 0 0 0	0 0 0 0 1 1 0 0 1 1 0 0	60	0 1 0 0 1 0 1 1	0 0 0 0 0 0 1 0 1 1 0 0
29	0 0 0 0 0 0 1 0	0 0 0 0 1 1 0 0 1 1 0 1	61	0 0 1 1 0 0 1 0	0 0 0 0 0 1 0 1 1 0 1 0
30	0 0 0 0 0 0 1 1	0 0 0 0 0 1 1 0 1 0 0 0	62	0 0 1 1 0 0 1 1	0 0 0 0 0 1 1 0 0 1 1 0
31	0 0 0 1 1 0 1 0	0 0 0 0 0 1 1 0 1 0 0 1	63	0 0 1 1 0 1 0 0	0 0 0 0 0 1 1 0 0 1 1 1

(b) Make-up Code (1/2)

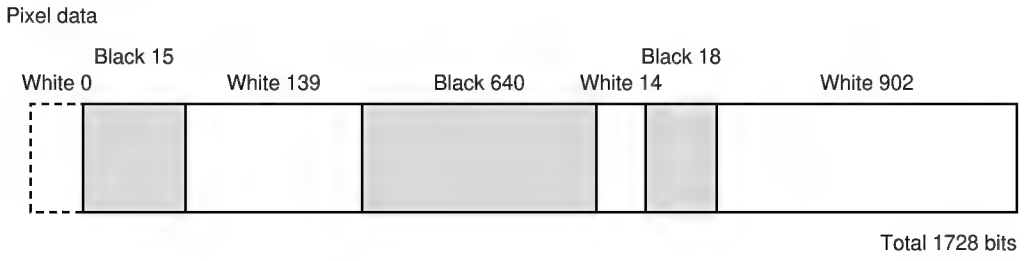
Run-Length	White Run Code Characters	Black Run Code Characters	Run-Length	White Run Code Characters	Black Run Code Characters
64	1 1 0 1 1	0 0 0 0 0 0 1 1 1 1	960	0 1 1 0 1 0 1 0 0	0 0 0 0 0 0 1 1 1 0 0 1 1
128	1 0 0 1 0	0 0 0 0 1 1 0 0 1 0 0 0	1024	0 1 1 0 1 0 1 0 1	0 0 0 0 0 0 1 1 1 0 1 0 0
192	0 1 0 1 1 1	0 0 0 0 1 1 0 0 1 0 0 1	1088	0 1 1 0 1 0 1 1 0	0 0 0 0 0 0 1 1 1 0 1 0 1
256	0 1 1 0 1 1 1	0 0 0 0 0 1 0 1 1 0 1 1	1152	0 1 1 0 1 0 1 1 1	0 0 0 0 0 0 1 1 1 0 1 1 0
320	0 0 1 1 0 1 1 0	0 0 0 0 0 0 1 1 0 0 1 1	1216	0 1 1 0 1 1 0 0 0	0 0 0 0 0 0 1 1 1 0 1 1 1
384	0 0 1 1 0 1 1 1	0 0 0 0 0 0 1 1 0 1 0 0	1280	0 1 1 0 1 1 0 0 1	0 0 0 0 0 0 1 0 1 0 0 1 0
448	0 1 1 0 0 1 0 0	0 0 0 0 0 0 1 1 0 1 0 1	1344	0 1 1 0 1 1 0 1 0	0 0 0 0 0 0 1 0 1 0 0 1 1
512	0 1 1 0 0 1 0 1	0 0 0 0 0 0 1 1 0 1 1 0 0	1408	0 1 1 0 1 1 0 1 1	0 0 0 0 0 0 1 0 1 0 1 0 0
576	0 1 1 0 1 0 0 0	0 0 0 0 0 0 1 1 0 1 1 0 1	1472	0 1 0 0 1 1 0 0 0	0 0 0 0 0 0 1 0 1 0 1 0 1
640	0 1 1 0 0 1 1 1	0 0 0 0 0 0 1 0 0 1 0 1 0	1536	0 1 0 0 1 1 0 0 1	0 0 0 0 0 0 1 0 1 1 0 1 0
704	0 1 1 0 0 1 1 0 0	0 0 0 0 0 0 1 0 0 1 0 1 1	1600	0 1 0 0 1 1 0 1 0	0 0 0 0 0 0 1 0 1 1 0 1 1
768	0 1 1 0 0 1 1 0 1	0 0 0 0 0 0 1 0 0 1 1 0 0	1664	0 1 1 0 0 0	0 0 0 0 0 0 1 1 0 0 1 0 0
832	0 1 1 0 1 0 0 1 0	0 0 0 0 0 0 1 0 0 1 1 0 1	1728	0 1 0 0 1 1 0 1 1	0 0 0 0 0 0 1 1 0 0 1 0 1
896	0 1 1 0 1 0 0 1 1	0 0 0 0 0 0 1 1 1 0 0 1 0	EOL	0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 1

(b) Make-up Code (2/2)

Run-Length	Code Characters (Common for White and Block Runs)	Run-Length	Code Characters (Common for White and Block Runs)
1792	0 0 0 0 0 0 0 1 0 0 0	2240	0 0 0 0 0 0 0 1 0 1 1 0
1856	0 0 0 0 0 0 0 1 1 0 0	2304	0 0 0 0 0 0 0 1 0 1 1 1
1920	0 0 0 0 0 0 0 1 1 0 1	2368	0 0 0 0 0 0 0 1 1 1 0 0
1984	0 0 0 0 0 0 0 1 0 0 1 0	2432	0 0 0 0 0 0 0 1 1 1 0 1
2048	0 0 0 0 0 0 0 1 0 0 1 1	2496	0 0 0 0 0 0 0 1 1 1 1 0
2112	0 0 0 0 0 0 0 1 0 1 0 0	2560	0 0 0 0 0 0 0 1 1 1 1 1
2176	0 0 0 0 0 0 0 1 0 1 0 1		

Figure 18-6 shows as example of the encoding of one line. If the first pixel of the line is black, the line will be encoded taking that there is 0 virtual white pixels immediately before the first pixel.

Figure 18-6. Encoding Example of One Line



Code data

White 0	Black 15	White 128	White 11	Black 640	Black 0	White 14	Black 18	White 896	White 6
00110101	000011000	10010	01000	0000001001010	0000110111	110100	0000001000	011010011	1110

Total 79 bits

The encoding of one page is performed using the following process.

<1> First the EOL (End Of Line) code in the code table is output.

<2> One line is encoded.

<3> If necessary, fill bits composed of several 0s are added to ensure minimum transmit time'.

<4> The EOL code is added.

<5> Steps <1> to <4> are repeated.

<6> The RTC (Return to Control) code is added after the code (or fill bit following the code) of the last line of one page and end of the page is shown. However:

RTC = EOL × 6

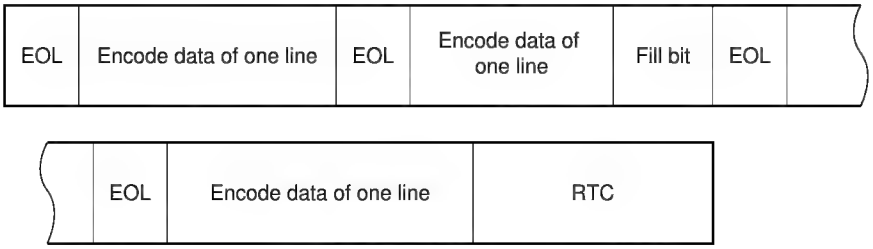
Note Minimum Transmit Time

For facsimiles, etc. there are various allowable receive speeds for the receiving side. The allowable receive speed is mainly determined by the size of the receive buffer and printout speed. The code data amount depends on the image data pattern and the transmit speed depends on the code data amount. Therefore, if short line codes are transmitted one after the other continuously even if the allowable receive speed is slow, the codes of the next line have to be received before images of one line have all been output, thus resulting in data loss.

For this reason, the minimum time (minimum transmit time) in which the code data of one line can be received is determined for each equipment. If the transmitting side transmits the code data of one line, and this time cannot be attained, insert fill bits and adjust. The minimum transmit time of the receiving side is notified to each equipment before the code data is transmitted.

The image data of one page is as shown in Figure 18-7 when encoded according to this process.

Figure 18-7. MH Encoding of One Page



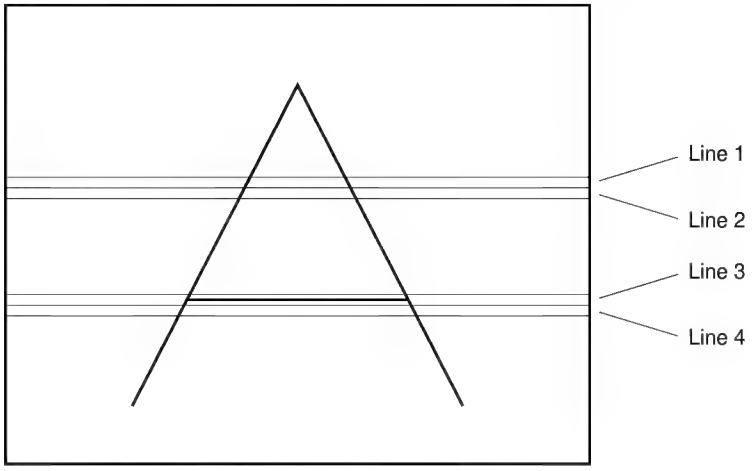
Remark RTC = EOL × 6

18.2.3 MR encoding

The MR encoding method focuses on the correlation between the vertical directions of the image data and encodes a line while comparing it with the line before.

A text image drawn with characters and diagrams is basically composed of vertical, horizontal, and diagonal lines. Therefore, if a certain image pattern appears for a certain line as lines 1 and 2 shown in Figure 18-8, it is highly possible that the same pattern will appear at the top and bottom lines as lines 3 and 4 shown in Figure 18-8. By assigning short codes to these two cases, a compression rate higher than that of MH encoding can be obtained.

Figure 18-8. Correlation Between Vertical Directions of Image



Before this method is explained, the terms and symbols used in the explanation are defined below.

The line immediately before the encoded line is called the reference line. The turning points on the encoded line and reference line are given labels a1, a2, b1 and b2 as shown in Figure 18-9. Three modes are considered from the positional relation between these four pixels and encoding is performed. a0 is the first pixel scanned and is called the reference point. The five pixels are defined as follows.

<1> a0: When a line is encoded, a virtual white pixels is assumed to exist immediately before the first pixel of the line and placed here. When a certain range of the line is encoded with a0 as the starting point, a new a0 position is fixed behind the encoded range.

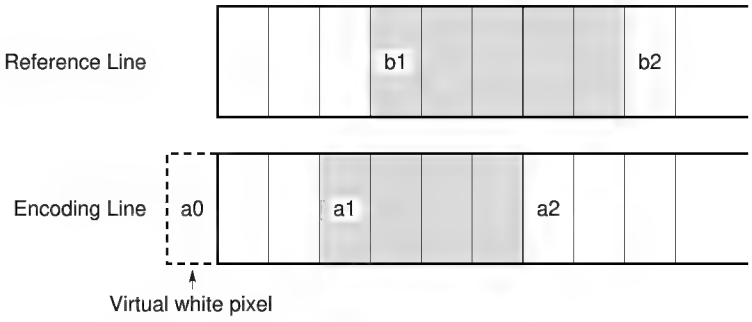
<2> a1: Placed at the turning point appearing after a0 on the encoded line.

<3> a2: Placed at the turning point appearing after a1 on the encoded line.

<4> b1: Placed at the first pixel of the opposite color to a0 appearing at the right of the a0 pixel on the reference line.

<5> b2: Placed at the turning point appearing next to b1 on the reference line.

Figure 18-9. Reference Points for Encoding



There are three encoding modes.

(1) Path mode
(2) Vertical mode
(3) Horizontal mode

They are described in detail below.

(1) Path mode

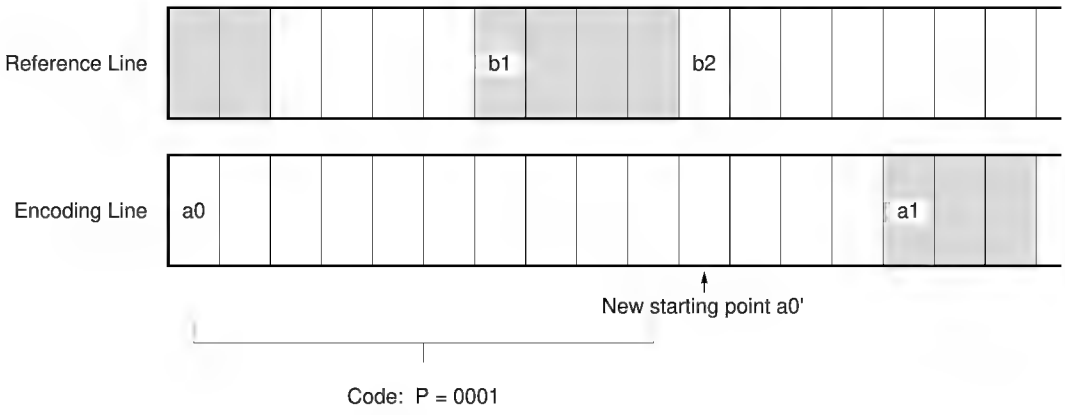
The path mode is used when b2 lies at the left side of a1. This means that the turning point after a0 on the encoding line does not lie before the position corresponding to b2 (Figure 18-10). This occurs in the combination of lines 3 and 4 shown in Figure 18-8.

Specifically, encoding is performed as follows.

<1> The path mode code P (=0001) is output as the code.

<2> A new reference point a0 is set at the position corresponding to b2. a0' becomes a0 by the following encoding.

Figure 18-10. Path Mode Example



(2) Vertical mode

The vertical mode is used when a1 lies within the 7 pixels to the right and left of b1 (Figure 18-11). This occurs in the combination of lines 1 and 2 shown in Figure 18-8.

Specifically, encoding is performed as follows.

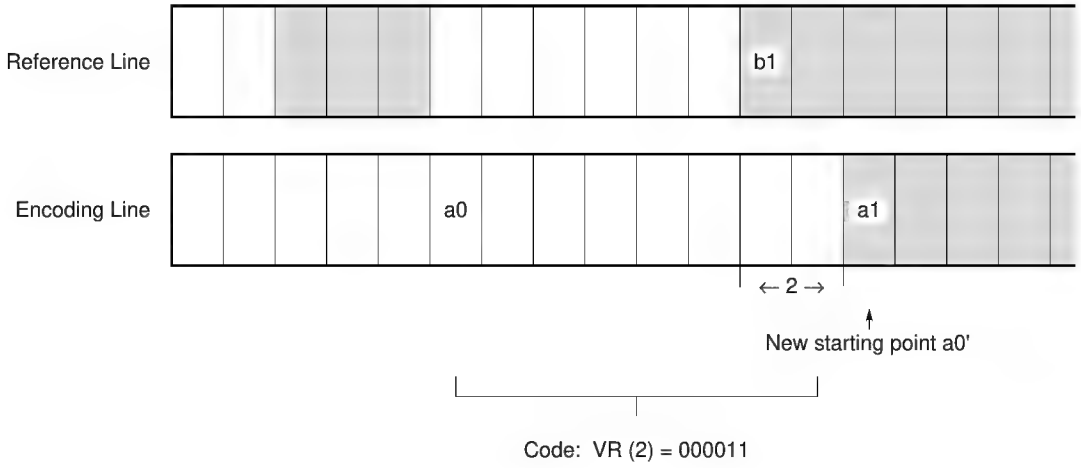
<1> Focusing on the correlative position of a1 (right or left) and distance (number of pixels) corresponding to b1, codes are output according to the table (Table 18-2).

<2> A new reference point a0' is set at the a1 position. a0' becomes a0 in encoding processes hereafter.

Table 18-2. Vertical Mode Code

Type	Positional Relation Between a1 and b1	Code
V (0)	a1 is directly below b1	1
VL (1)	a1 is at the 1st pixel to the left of b1	010
VL (2)	a1 is at the 2nd pixel to the left of b1	000010
VL (3)	a1 is at the 3rd pixel to the left of b1	0000010
VR (1)	a1 is at the 1st pixel to the right of b1	011
VR (2)	a1 is at the 2nd pixel to the right of b1	000011
VR (3)	a1 is at the 3rd pixel to the right of b1	0000011

Figure 18-11. Vertical Mode Example (When a1 lies at the 2nd pixel to the right of b1)



(3) Horizontal mode

When the path mode or not the vertical mode (when b2 is at the right side of a1), and the distance between a1 and b1 exceeds 3 pixels, this mode is used (Figure 18-12).

Specifically, encoding is performed as follows.

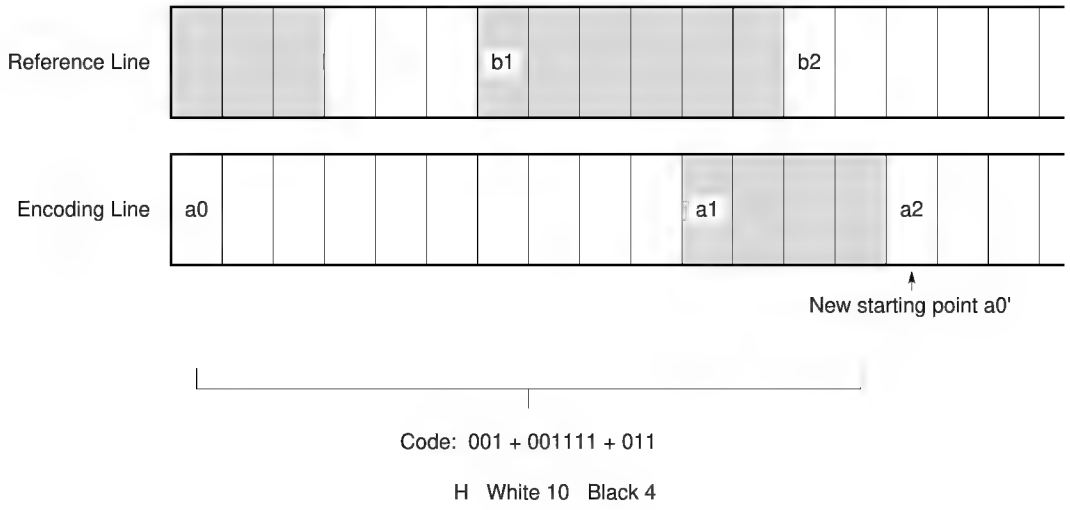
<1> First the horizontal mode code H (=001) is output.

<2> The run-length between a0 and a1 is encoded according to the MH encoding code table (Table 18-1).

<3> The run-length between a1 and a2 is encoded according to the MH encoding code table (Table 18-1).

<4> A new reference point a0' is set at the a2 position. a0' becomes a0 in encoding processes hereafter.

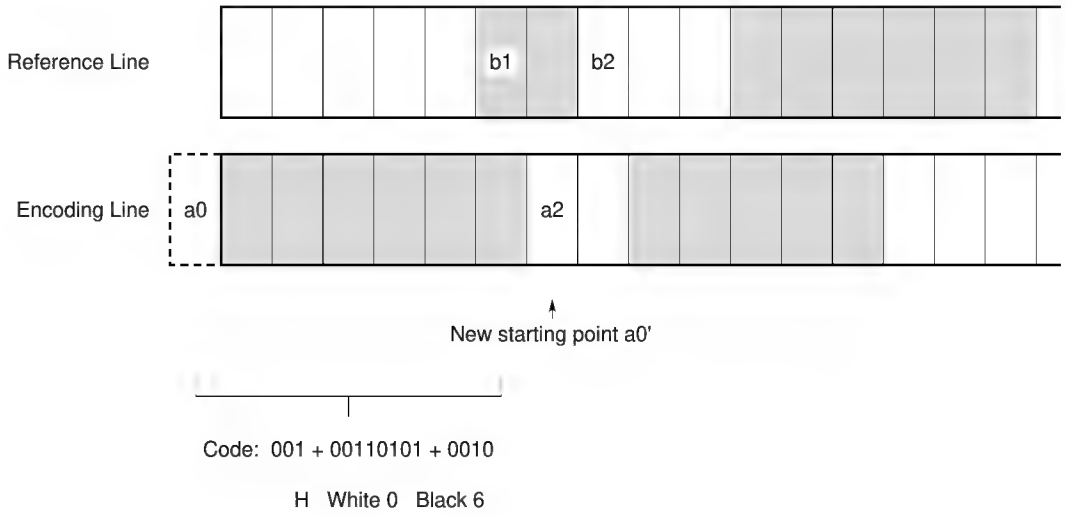
Figure 18-12. Horizontal Mode Example



Special processes must be performed because the a0 through a2 and b1 and b2 relations are special at the starting and ending points of each line.

a0 is placed at the starting point of a line assuming that there is 0 virtual white pixel immediately before the first pixel of the line. Therefore, if a line starts with a black pixel as shown in Figure 18-13, the run-length code 0 lies after code H.

Figure 18-13. Processing Example of Starting Point of Line



At the ending point of the line, 0 virtual pixel of the opposite color of the last pixel are assumed to follow the last pixel and the line is encoded until the virtual pixels (Figure 18-14).

Figure 18-14. Processing Example of Ending Point of Line

The above process is called the two-dimensional encoding method.

The encoding of one page is performed by combining the one-dimensional and two-dimensional encoding methods and using the parameter K (Figure 18-15).

Specifically, encoding is performed as follows.

<1> The 1st line is one-dimensional encoded.

<2> Lines after the 2nd line are two-dimensional encoded.

<3> K + 1st line is one-dimensional encoded.

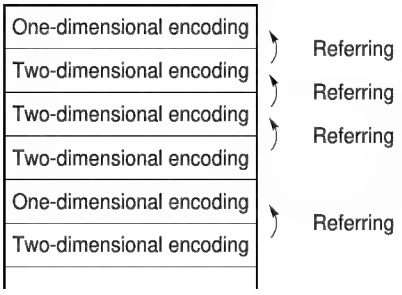
<4> Steps <1> to <3> are repeated.

This means that every K lines are encoded by one-dimensional encoding. This is because two-dimensional encoding is performed by referring to the line directly before, and therefore, if a certain line could not be encoded normally due to transmission errors, the decoded images after that line will all be abnormal. To prevent this, one-dimensional encoding is performed for every K lines.

The K value is generally as follows according to the resolution in the vertical direction.

- When 100LPI (Lines Per Inch), K = 2
- When 200LPI (Lines Per Inch), K = 4

Figure 18-15. K Parameter (When K = 4)



To differentiate between one-dimensional encoded lines and two-dimensional encoded lines, division codes of the line codes are added.

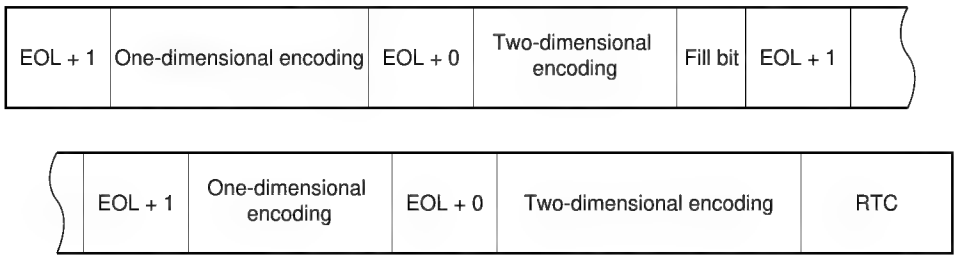
- Directly before one-dimensional codes: EOL + 1
- Directly before two-dimensional codes: EOL + 0

The following code is also added to the ending point of a page.

RTC = (EOL + 1) × 6

When encoded according to this process, the image data of one page becomes as shown in Figure 18-16.

Figure 18-16. MR Encoding of One Page



Remark RTC = (EOL + 1) × 6

18.2.4 MMR encoding

The MMR encoding method two-dimensional encodes all lines. The 1st line of a page is two-dimensional encoded assuming that there is a virtual all white line directly before the 1st line of a page.

The MR method is established on the basis that encoded data will be transmitted via normal telephone lines. K parameters are provided to limit the reception of transmission errors. On the other hand, as the MMR encoding method is performed on the basis that data will be transmitted error-free using digital lines, the K parameter is set infinitely. The encoding of one page is performed as follows.

<1> Assuming that there is a virtual all white line directly before the 1st line of a page, all the lines from the 1st line are two-dimensional encoded. No special codes to divide each line code are used.

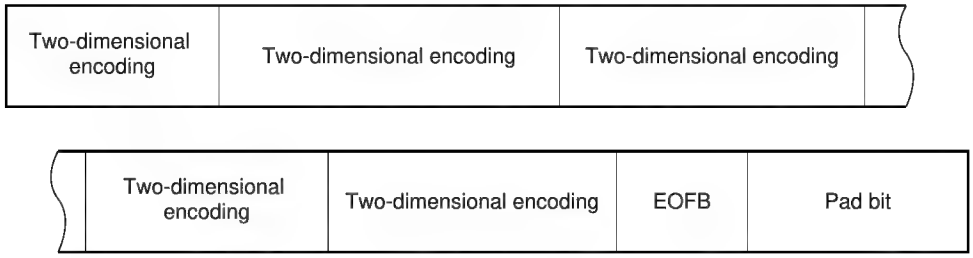
<2> The EOFB (End of Facsimile Block) code is added immediately after the code of the last line to indicate the ending point of the page. The EOFB value is as follows.

EOFB = EOL × 2

<3> If the encoded data of one page must be arranged in units of byte, or arranged more than a certain size, 0 bit rows are added after the EOFB code as pad bits.

When encoded according to this process, the image data of one page becomes as shown in Figure 18-17.

Figure 18-17. MMR Encoding of One Page



Remark EOFB = EOL × 2

18.3 Memory Map

The V55PI code instructions require the following data memory areas.

(1) Register file area

This is a parameter setting register bank.

Refer to **18.6 Descriptions of Functions of Each Instruction** for details of parameter setting.

(2) User RAM

Encoding line turning point table	:	Area for storage of turning point information necessary for encoding (for nbit/line, a maximum area of 2n + 4 bytes is required)
Reference line table	:	Area for storage of reference line turning point information
Pixel data buffer	:	Area for storage of pixel data read from the scanner in the case of encoding and area for storage of encoded data received from modem
Transmit/receive buffer	:	Buffer to transfer the encoded data to the modem/scanner
Print buffer	:	Buffer to transfer the decoded pixel data to the recording system

(3) User ROM

Encoding conversion table	:	Conversion table for MH and MR encoding
Decoding conversion table	:	Conversion table for MH and MR encoding

(4) Accessing expansion memory space

16 Mbyte expansion memory spaces can be accessd using the expansion segment override prefix instruction (DS2: or DS3:)

The DS2 and DS3 in the register bank for the parameter setting of each instruction are used as the segment register DS2 and DS3 for execuitng the instructions.

Table 18-3. Codec Instructions Addable with Expansion Segment Override Prefix

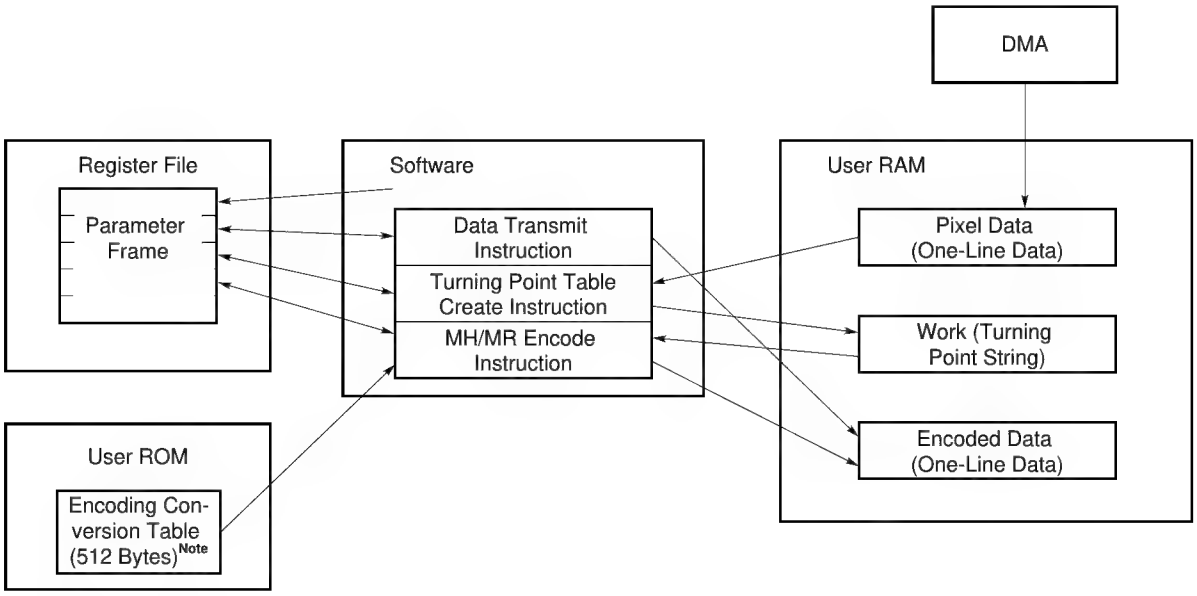
DS2:	DS3:	Codec Instruction
Yes	Yes	COLTRP
Yes	No	MHENC
Yes	Yes	MHDEC
Yes	No	MRENC
Yes	Yes	MRDEC
Yes	No	SCHEOL
Yes	No	GETBIT
Yes	Yes	CNVTRP

Description example

```
DS2: DS3: COLTRP
DS2:  SCHEOL
```

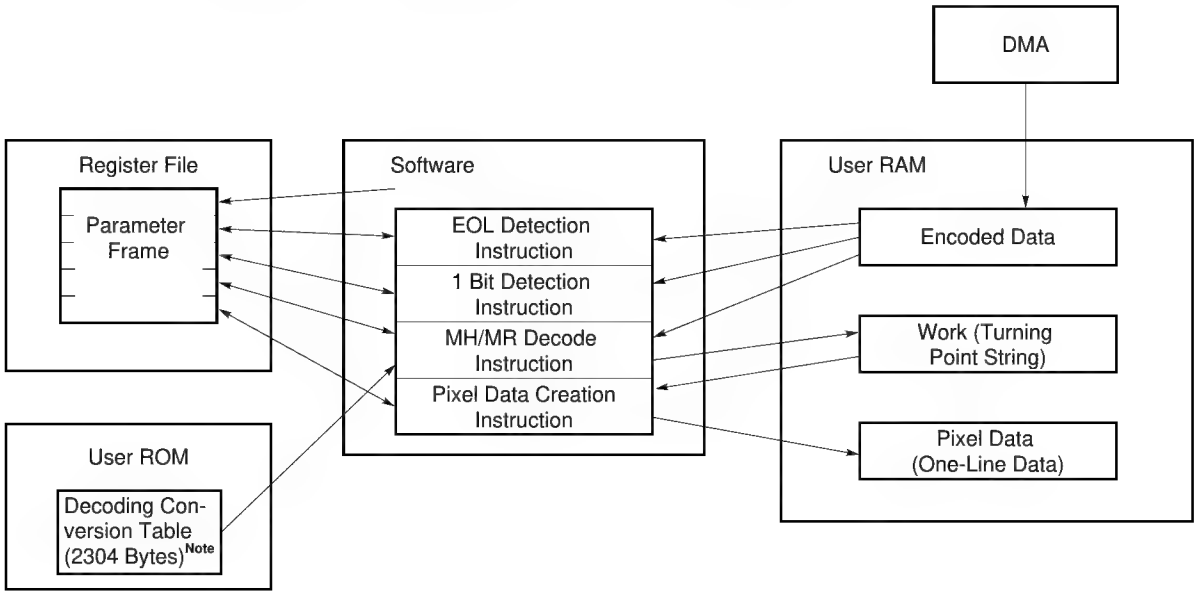
Figure 18-18 shows relations between encode instructions and data in each memory. Figure 18-19 shows relations between decode instructions and data in each memory.

Figure 18-18. Encode Instructions and Data in Each Memory



Note MH, MR encode instructions.

Figure 18-19. Decode Instructions and Data in Each Memory



Note MH, MR decode instructions.

18.4 Encoding/Decoding Conversion Tables

The encoding conversion table is used to convert pixel data (run-length) to MH codes. When using the MH encoding instruction "MHENC" and the MR encoding instruction "MRENC", specify the table which is specified in **18.4.1 Encoding Conversion Table** (see **18.4.3 Encoding conversion table (for MH encoding instruction)**).

The decoding conversion table is used to convert codes to pixel data (run-length). When using the MH decoding instruction "MHDEC" and the MR decoding instruction "MRDEC", specify the table which is specified in **18.4.2 Decoding conversion table** (see **18.4.3 Decoding conversion table (MH decoding instruction)**).

The start offset of the both encoding/decoding conversion tables should be allocated at address 0000H.

18.4.1 Encoding conversion table

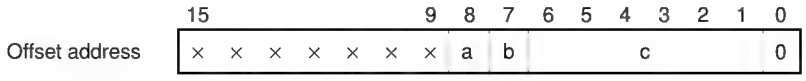
The encoding conversion table is configured according to the following rule.

The run-length = R pixel data code (MH) is stored in the offset address specification destination of the encoding conversion table as shown below.

(1) Offset address

The offset address is generated as follows.

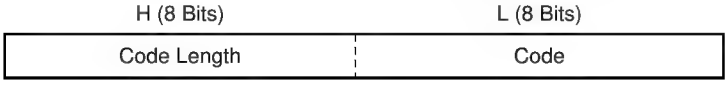
Example When R = 64M + T



a: 0 = white, 1 = black
b: 0 = terminate code, 1 = make-up code
c: when terminate code (b = 0)
T (binary) value
when make-up code (b = 1)
M (binary) value

(2) Offset address specification destination

The following code is stored in the word area specified by the offset address.



- When a code is less than 8 bits, it is right-justified, and the remaining bits are set to 0.
- When a code is 8 bits or more, it is left-justified.
- A code is indicated LSB-first.

Example When white run-length R = 708

$R = 64 \times 11 + 4 \rightarrow M = 11, T = 4$

- Make-up code
  The code is "0**11001100**".
  This is stored in the word area indicated by offset address = xxxxxxx010010110 as shown below. All that are not stored in this area are set to 0.

0 0 0 0 1 0 0 1	**0 0 1 1 0 0 1 1**
Code Length	Code

- Terminate code
  The code is " **1011** ".
  This is stored in the word area indicated by offset address = xxxxxxx000001000 as shown below. The remaining bits are all set to 0.

0 0 0 0 0 1 0 0	0 0 0 0 **1 1 0 1**
Code Length	Code

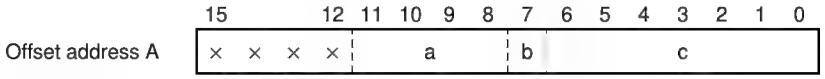
18.4.2 Decoding conversion table

The decoding conversion table is generated according to the following rule.

The number of consecutive "0"s of the code data to be converted is counted.

(1) Offset address A

Offset address A is generated as follows.

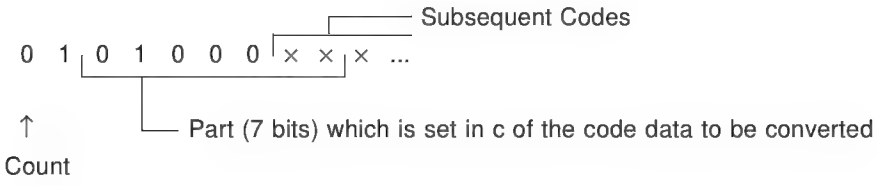


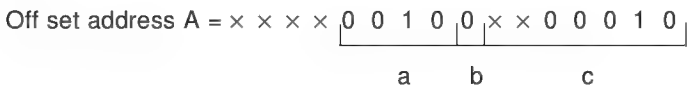
a: (Number of counted "0"s) + 1

b: 0 = white, 1 = black

c: Code data to be converted

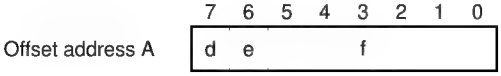
Example When code to be converted = 0101000 (white run-length = 24)





(2) Offset address A specification destination

The following data is stored in the byte area specified by offset address A.



d: 0 = white, 1 = black

e: 0 = terminate code, 1 = make-up code

f: run-length

(3) Offset address B

The 12 bits resulting from the addition of the 8-bit data stored in (2) and "000H" is referred to as offset address B. In the byte area specified by offset address B, "code length - (number of counted "0"s + 1)" of the code to be converted is stored.

Example Offset address B = ××××000000011000(×018H)
(when the code to be converted is "01<u>01000</u>")

	7	6	5	4	3	2	1	0	
Offset address B	0	0	0	0	0	1	0	1	7 – (1 + 1) = 5

18.4.3 Decoding conversion table (for MH decoding instruction)

```
;*****************************************
;*     MH decode (expansion) le              *
;*        compression code → data            *
;*****************************************
;
;       * EOL
;          Number of '0' = 11
;
;       * EOL + '1'
;       -----4bit---+-1-+-------------7bit---------------
;       +---+---+---+---+---+---+---+---+---+---+---+---+
;       |num of '0'+1|col|           search data          | ---+
;       +---+---+---+---+---+---+---+---+---+---+---+---+    |
;           num of '0'+1: number of '0'+1                    |
;           col: color                                       |
;           search data: 7 bits                              |
;           ex) 0001 1011xxxx                                |
;               num of '0' = 3                               |
;               search data = 1011xxx                        |
;                                                            |
;       --1-+-1-+---------6bit-----------                    |
;       +---+---+---+---+---+---+---+---+                    |
;       |col|t/m|        run length       | <------------------+
;       +---+---+---+---+---+---+---+---+ -------------+
;           col: color                                   |
;           t/m: 0 = terminate code                      |
;                1 = makeup code                         |
;           run length: 6 bits                           |
;                                                        |
;       ------4bit------+-----4bit-------                |
;       +---+---+---+---+---+---+---+---+                |
;       | 0   0   0   0 |sch data length| <------------+
;       +---+---+---+---+---+---+---+---+
;        data length
;           terminate: number of '0'+1 + search data length
;           makeup   : (number of '0'+1 + search data length)*64
```

```
;
;+++++++++++++ Decode Table +++++++++++++
code    segment at      xxxxh                       ;    offset
        ORG     0h
;------ White Terminal Data Table -------
        db      05h,02h,02h,03h,03h,03h,03h,03h   ; W.T=000h-007h
        db      04h,04h,02h,03h,03h,01h,05h,05h   ; W.T=008h-00Fh
        db      05h,05h,05h,03h,03h,04h,01h,02h   ; W.T=010h-017h
        db      05h,05h,04h,05h,04h,01h,01h,04h   ; W.T=018h-01Fh
        db      04h,04h,04h,04h,04h,04h,04h,05h   ; W.T=020h-
        db      05h,05h,05h,05h,05h,02h,02h,03h
        db      03h,06h,06h,06h,06h,05h,05h,06h   ; W.T=030h-
        db      06h,06h,06h,06h,06h,05h,05h,05h
;------- White Makeup Data Table --------
        db      00h,04h,04h,04h,05h,05h,05h,06h   ; W.M=040h-
        db      06h,06h,06h,07h,07h,07h,07h,07h
        db      07h,07h,07h,07h,07h,07h,07h,07h   ; W.M=050h-057h
        db      07h,07h,04h,07h                   ; W.M=058h-05Bh
;---- White/Black Addition Data Table ---
        db      03h,03h,03h,04h                   ; W/B.M=05Ch-05Fh
        db      04h,04h,04h,04h,04h,04h,04h,04h   ; W/B.M=060h-067h
        db      04h,00h,00h,00h,00h,00h,00h,00h   ; W/B.M=068h dummy=069h-
        db      00h,00h,00h,00h,00h,00h,00h,00h   ; dummy
        db      00h,00h,00h,00h,00h,00h,00h,00h
;------ Black Terminate Data Table ------
        db      05h,01h,01h,01h,01h,01h,01h,01h   ; B.T=080h-087h
        db      02h,02h,02h,02h,02h,02h,02h,04h   ; B.T=088h-08Fh
        db      04h,04h,03h,06h,06h,06h,05h,05h   ; B.T=090h-
        db      04h,04h,07h,07h,07h,07h,06h,06h
        db      06h,06h,07h,07h,07h,07h,07h,07h   ; B.T=0A0h-
        db      06h,06h,07h,07h,06h,06h,06h,06h
        db      06h,06h,06h,06h,05h,05h,05h,05h   ; B.T=0B0h-
        db      05h,06h,06h,05h,05h,06h,06h,06h
;------- Black Makeup Data Table --------
        db      00h,03h,07h,07h,06h,05h,05h,05h   ; B.M=0C0h-
        db      06h,06h,06h,06h,06h,06h,06h,06h
        db      06h,06h,06h,06h,06h,06h,06h,06h   ; B.M=0D0h-0D7h
        db      06h,06h,06h,06h                   ; B.M=0D8h-0DBh
```

```
        ORG     0100h
;+++++++++++++ Decode Table +++++++++++++
        db      003h,005h,009h,006h,042h,00Eh,004h,007h     ; address=0100h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h     ; address=0110h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h     ; address=0120h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h     ; address=0130h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h     ; address=0140h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h     ; address=0150h-
        db      003h,005h,011h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Eh,004h,007h     ; address=0160h-
        db      003h,005h,010h,006h,008h,041h,004h,007h
        db      003h,005h,009h,006h,042h,00Fh,004h,007h     ; address=0170h-
        db      003h,005h,011h,006h,008h,041h,004h,007h

        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=0180h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=0190h-
        db      083h,082n,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01A0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01B0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01C0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01D0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01E0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
        db      083h,082h,083h,082h,083h,082h,083h,082h     ; address=01F0h-
        db      083h,082h,083h,082h,083h,082h,083h,082h
```

```
        ORG     0200h
;+++++++++++++ Decode Table +++++++++++++
        db      00Bh,05Ah,018h,002h,01Bh,049h,037h,002h     ; address=0200h-
        db      00Bh,047h,033h,002h,057h,053h,043h,002h
        db      00Bh,05Ah,031h,002h,03Bh,04Fh,039h,002h     ; address=0210h-
        db      00Bh,04Bh,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,04Dh,038h,002h     ; address=0220h-
        db      00Bh,048h,034h,002h,059h,055h,043h,002h
        db      00Bh,05Ah,032h,002h,03Ch,051h,03Ah,002h     ; address=0230h-
        db      00Bh,04Ah,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,049h,037h,002h     ; address=0240h-
        db      00Bh,047h,033h,002h,058h,054h,043h,002h
        db      00Bh,05Ah,031h,002h,03Bh,050h,039h,002h     ; address=0250h-
        db      00Bh,04Ch,019h,002h,012h,044h,043h,002h
        db      00Bh,05Ah,018h,002h,01Bh,04Eh,038h,002h     ; address=0260h-
        db      00Bh,048h,034h,002h,05Bh,056h,043h,002h
        db      00Bh,05Ah,032h,002h,03Ch,052h,03Ah,002h     ; address=0270h-
        db      00Bh,04Ah,019h,002h,012h,044h,043h,002h

        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=0280h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=0290h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02A0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02B0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02C0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02D0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02E0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
        db      081h,084h,081h,084h,081h,084h,081h,084h     ; address=02F0h-
        db      081h,084h,081h,084h,081h,084h,081h,084h
```

```
        ORG     0300h
;+++++++++++++ Decode Table ++++++++++++
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah     ; address=0300h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah     ; address=0310h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah     ; address=0320h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah     ; address=0330h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah     ; address=0340h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah     ; address=0350h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah
        db      00Ch,01Ch,027h,00Ah,035h,03Fh,02Bh,00Ah     ; address=0360h-
        db      00Ch,03Dh,029h,00Ah,01Ah,045h,015h,00Ah
        db      00Ch,01Ch,028h,00Ah,036h,000h,02Ch,00Ah     ; address=0370h-
        db      00Ch,03Eh,02Ah,00Ah,01Ah,046h,015h,00Ah

        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=0380h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=0390h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03A0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03B0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03C0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03D0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03E0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
        db      086h,085h,086h,085h,086h,085h,086h,085h     ; address=03F0h-
        db      086h,085h,086h,085h,086h,085h,086h,085h
```

```
        ORG     0400h
;++++++++++++++ Decode Table ++++++++++++++
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0400h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0410h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0420h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0430h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0440h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0450h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0460h-
        db      014h,013h,024h,001h,022h,020h,026h,001h
        db      014h,013h,023h,001h,021h,01Fh,025h,001h     ; address=0470h-
        db      014h,013h,024h,001h,022h,020h,026h,001h

        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=0480h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=0490h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04A0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04B0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04C0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04D0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04E0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
        db      089h,087h,088h,087h,089h,087h,088h,087h     ; address=04F0h-
        db      089h,087h,088h,087h,089h,087h,088h,087h
```

```
        ORG     0500h
;+++++++++++++ Decode Table +++++++++++++
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0500h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0510h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0520h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0530h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0540h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0550h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0560h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh     ; address=0570h-
        db      017h,00Dh,02Fh,00Dh,017h,00Dh,030h,00Dh

        db      08Ah,08Fh,08Bh,08Ch,08Ah,094h,08Bh,08Ch     ; address=0580h-
        db      08Ah,0C2h,08Bh,08Ch,08Ah,095h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A4h,08Bh,08Ch     ; address=0590h-
        db      08Ah,09Ch,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A2h,08Bh,08Ch     ; address=05A0h-
        db      08Ah,09Ah,08Bh,08Ch,08Ah,0AAh,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A6h,08Bh,08Ch     ; address=05B0h-
        db      08Ah,093h,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,094h,08Bh,08Ch     ; address=05C0h-
        db      08Ah,0C3h,08Bh,08Ch,08Ah,095h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A5h,08Bh,08Ch     ; address=05D0h-
        db      08Ah,09Dh,08Bh,08Ch,08Ah,080h,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A3h,08Bh,08Ch     ; address=05E0h-
        db      08Ah,09Bh,08Bh,08Ch,08Ah,0ABh,08Bh,08Ch
        db      08Ah,08Fh,08Bh,08Ch,08Ah,0A7h,08Bh,08Ch     ; address=05F0h-
        db      08Ah,093h,08Bh,08Ch,08Ah,080h,08Bh,08Ch
```

```
        ORG     0600h
;+++++++++++++ Decode Table +++++++++++++
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0600h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0610h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0620h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=062Eh-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0640h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0650h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0660h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h     ; address=0670h-
        db      02Dh,016h,02Eh,016h,02Dh,016h,02Eh,016h

        db      08Dh,091h,097h,08Eh,08Dh,09Eh,0B9h,08Eh     ; address=0680h-
        db      08Dh,0B0h,0ACh,08Eh,08Dh,0A8h,090h,08Eh
        db      08Dh,091h,0B2h,08Eh,08Dh,0A0h,0BDh,08Eh     ; address=0690h-
        db      08Dh,0BEh,0AEh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Fh,0BAh,08Eh     ; address=06A0h-
        db      08Dh,0B1h,0ADh,08Eh,08Dh,0A9h,090h,08Eh
        db      08Dh,091h,0B3h,08Eh,08Dh,0A1h,0C4h,08Eh     ; address=06B0h-
        db      08Dh,0BFh,0AFh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Eh,0B9h,08Eh     ; address=06C0h-
        db      08Dh,0B0h,0ACh,08Eh,08Dh,0A8h,090h,08Eh
        db      08Dh,091h,0B2h,08Eh,08Dh,0A0h,0BDh,08Eh     ; address=06D0h-
        db      08Dh,0BEh,0AEh,08Eh,08Dh,096h,090h,08Eh
        db      08Dh,091h,097h,08Eh,08Dh,09Fh,0BAh,08Eh     ; address=06E0h-
        db      08Dh,0B1h,0ADh,08Eh,08Dh,0A9h,090h,08Eh
        db      08Dh,091h,0B3h,08Eh,08Dh,0A1h,0C4h,08Eh     ; address=06F0h-
        db      08Dh,0BFh,0AFh,08Eh,08Dh,096h,090h,08Eh
```

```
        ORG     0700h
;+++++++++++++ Decode Table ++++++++++++
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0700h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0710h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0720h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=072Eh-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0740h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0750h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0760h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh     ; address=0770h-
        db      01Dh,01Eh,01Dh,01Eh,01Dh,01Eh,01Dh,01Eh

        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h     ; address=0780h-
        db      092h,0DAh,0D6h,0D0h,0CCh,0C8h,098h,0C1h
        db      092h,099h,0D4h,0CEh,0CAh,0C7h,0D8h,0C1h     ; address=0790h-
        db      092h,0C5h,0BBh,0D2h,0B7h,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h     ; address=07A0h-
        db      092h,0DBh,0D7h,0D1h,0CDh,0C9h,098h,0C1h
        db      092h,099h,0D5h,0CFh,0CBh,0C7h,0D9h,0C1h     ; address=07B0h-
        db      092h,0C5h,0BBh,0D3h,0B7h,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h     ; address=07C0h-
        db      092h,0DAh,0D6h,0D0h,0CCh,0C8h,098h,0C1h
        db      092h,099h,0D4h,0CEh,0CAh,0C7h,0D8h,0C1h     ; address=07D0h-
        db      092h,0C5h,0BBh,0D2h,0B7b,0B5h,098h,0C1h
        db      092h,099h,0B8h,0B6h,0B4h,0C6h,0BCh,0C1h     ; address=07E0h-
        db      092h,0DBh,0D7h,0D1h,0CDh,0C9h,098h,0C1h
        db      092h,099h,0D5h,0CFh,0CBh,0C7h,0D9h,0C1h     ; address=07F0h-
        db      092h,0C5h,0BBh,0D3h,0B7h,0B5h,098h,0C1h
```

```
        ORG     0800h
;+++++++++++++ Decode Table +++++++++++++
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0800h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0810h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0820h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0830h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0840h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0850h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0860h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0870h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
;
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0880h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=0890h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08A0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08B0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08C0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08D0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08E0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
        db      05Ch,05Dh,061h,065h,05Fh,05Eh,063h,067h     ; address=08F0h-
        db      05Ch,05Dh,062h,066h,060h,05Eh,064h,068h
code    ends

        end
```

18.4.4 Encoding conversion table (for MH encoding instruction)

```
;*****************************************
;*      MH encode (compression) table       *
;*         data → compression code          *
;*****************************************
;
;       address= +XXXXh
;
;       -1+1+----6bit---+1-      -----8bit(H)----+----8bit(L)-----
;       +-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;       |a|b|     c     |0| ===> |  code length  |               |
;       +-+-+-+-+-+-+-+-+-+-+      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
;
;       * Run Length >= 64
;        Run_Length = 64T +T
;        a: 0 = white
;           1 = black
;        b: 0 = terminate --- c = T
;           1 = makeup    --- c = M
;
;       * Run Length < 64
;        Only change the terminate code
;
code    segment at      xxxxh
        ORG     0h
;++++++++++ White Encode Table ++++++++++
;-------------- Terminate ---------------
        db      0ACh, 008h              ; address= +0000h
        db      038h, 006h
        db      00Eh, 004h
        db      001h, 004h
        db      00Dh, 004h
        db      003h, 004h
        db      007h, 004h
        db      00Fh, 004h
        db      019h, 005h              ; address= +0010h
        db      005h, 005h
        db      01Ch, 005h
        db      002h, 005h
        db      004h, 006h
        db      030h, 006h
```

```
db      00Bh, 006h
db      02Bh, 006h
db      015h, 006h              ; address= +0020h
db      035h, 006h
db      072h, 007h
db      018h, 007h
db      008h, 007h
db      074h, 007h
db      060h, 007h
db      010h, 007h
db      00Ah, 007h              ; address= +0030h
db      06Ah, 007h
db      064h, 007h
db      012h, 007h
db      00Ch, 007h
db      040h, 008h
db      0C0h, 008h
db      058h, 008h
db      0D8h, 008h              ; address= +0040h
db      048h, 008h
db      0C8h, 008h
db      028h, 008h
db      0A8h, 008h
db      068h, 008h
db      0E8h, 008h
db      014h, 008h
db      094h, 008h              ; address= +0050h
db      054h, 008h
db      0D4h, 008h
db      034h, 008h
db      0B4h, 008h
db      020h, 008h
db      0A0h, 008h
db      050h, 008h
db      0D0h, 008h              ; address= +0060h
db      04Ah, 008h
db      0CAh, 008h
db      02Ah, 008h
db      0AAh, 008h
```

```
        db      024h, 008h
        db      0A4h, 008h
        db      01Ah, 008h
        db      09Ah, 008h              ; address= +0070h
        db      05Ah, 008h
        db      0DAh, 008h
        db      052h, 008h
        db      0D2h, 008h
        db      04Ch, 008h
        db      0CCh, 008h
        db      02Ch, 008h
;-------------- Terminate ---------------
        db      000h, 000h              ; address= +0080h / no data
        db      01Bh, 005h
        db      009h, 005h
        db      03Ah, 006h
        db      076h, 007h
        db      06Ch, 008h
        db      0ECh, 008h
        db      026h, 008h
        db      0A6h, 008h              ; address= +0090h
        db      016h, 008h
        db      0E6h, 008h
        db      033h, 009h
        db      0B3h, 009h
        db      04Bh, 009h
        db      0CBh, 009h
        db      02Bh, 009h
        db      0ABn, 009h              ; address= + 00A0h
        db      06Bh, 009h
        db      0EBh, 009h
        db      01Bh, 009h
        db      09Bh, 009h
        db      05Bh, 009h
        db      0DBh, 009h
        db      019h, 009h
        db      099h, 009h              ; address= +00B0h
        db      059h, 009h
        db      006h, 006h
```

```
        db      0D9h, 009h
        db      010h, 00Bh
        db      030h, 00Bh
        db      0B0h, 00Bh
        db      048h, 00Ch
        db      0C8h, 00Ch              ; address= +00C0h
        db      028h, 00Ch
        db      0A8h, 00Ch
        db      068h, 00Ch
        db      0E8h, 00Ch
        db      038h, 00Ch
        db      0B8h, 00Ch
        db      078h, 00Ch
        db      0F8h, 00Ch              ; address=  +00D0h

;++++++++++ Black Encode TAble ++++++++++
        ORG     0100h
;-------------- Terminate ---------------
        db      0ECh, 00Ah              ; address= +0100h
        db      002h, 003h
        db      003h, 002h
        db      001h, 002h
        db      006h, 003h
        db      00Ch, 004h
        db      004h, 004h
        db      018h, 005h
        db      028h, 006h              ; address= +0110h
        db      008h, 006h
        db      010h, 007h
        db      050h, 007h
        db      070h, 007h
        db      020h, 008h
        db      0E0h, 008h
        db      018h, 009h
        db      0E8h, 00Ah              ; address= +0120h
        db      018h, 00Ah
        db      010h, 00Ah
        db      0E6h, 00Bh
        db      016h, 00Bh
```

```
db      036h, 00Bh
db      0ECh, 00Bh
db      014h, 00Bh
db      0E8h, 00Bh              ; address= +0130h
db      018h, 00Bh
db      053h, 00Ch
db      0D3h, 00Ch
db      033h, 00Ch
db      0B3h, 00Ch
db      016h, 00Ch
db      096h, 00Ch
db      056h, 00Ch              ; address= +0140h
db      0D6h, 00Ch
db      04Bh, 00Ch
db      0CBh, 00Ch
db      02Bh, 00Ch
db      0ABh, 00Ch
db      06Bh, 00Ch
db      0EBh, 00Ch
db      036h, 00Ch              ; address= +0150h
db      0B6h, 00Ch
db      05Bh, 00Ch
db      0DBh, 00Ch
db      02Ah, 00Ch
db      0AAh, 00Ch
db      06Ah, 00Ch
db      0EAh, 00Ch
db      026h, 00Ch              ; address= +0160h
db      0A6h, 00Ch
db      04Ah, 00Ch
db      0CAh, 00Ch
db      024h, 00Ch
db      0ECh, 00Ch
db      01Ch, 00Ch
db      0E4h, 00Ch
db      014h, 00Ch              ; address= +0170h
db      01Ah, 00Ch
db      09Ah, 00Ch
db      0D4h, 00Ch
```

```
        db      034h, 00Ch
        db      05Ah, 00Ch
        db      066h, 00Ch
        db      0E6h, 00Ch
;-------------- Makeup ------------------
        db      000h, 000h              ; address= +0180h / no data
        db      0F0h, 00Ah
        db      013h, 00Ch
        db      093h, 00Ch
        db      0DAh, 00Ch
        db      0CCh, 00Ch
        db      02Ch, 00Ch
        db      0ACh, 00Ch
        db      036h, 00Dh              ; address= +0190h
        db      0B6h, 00Dh
        db      052h, 00Dh
        db      0D2h, 00Dh
        db      032h, 00Dh
        db      0B2h, 00Dh
        db      04Eh, 00Dh
        db      0CEh, 00Dh
        db      02Eh, 00Dh              ; address= +01A0h
        db      0AEh, 00Dh
        db      06Eh, 00Dh
        db      0EEh, 00Dh
        db      04Ah, 00Dh
        db      0CAh, 00Dh
        db      02Ah, 00Dh
        db      0AAh, 00Dh
        db      05Ah, 00Dh              ; address= +01B0h
        db      0DAh, 00Dh
        db      026h, 00Dh
        db      0A6h, 00Dh
        db      010h, 00Bh
        db      030h, 00Bh
        db      0B0h, 00Bh
        db      048h, 00Ch
        db      0C8h, 00Ch              ; address= +01C0h
        db      028h, 00Ch
```

```
        db      0A8h, 00Ch
        db      068h, 00Ch
        db      0E8h, 00Ch
        db      038h, 00Ch
        db      0B8h, 00Ch
        db      078h, 00Ch
        db      0F8h, 00Ch              ; address= +01D0h
code    ends
        end
```

18.5 Processing Flow

In encoding/decoding, the dedicated instructions described in **18.1 Features** is used in the order shown in Figures 18-20 and 18-21.

Figure 18-20. One-Line Data Encoding Flow

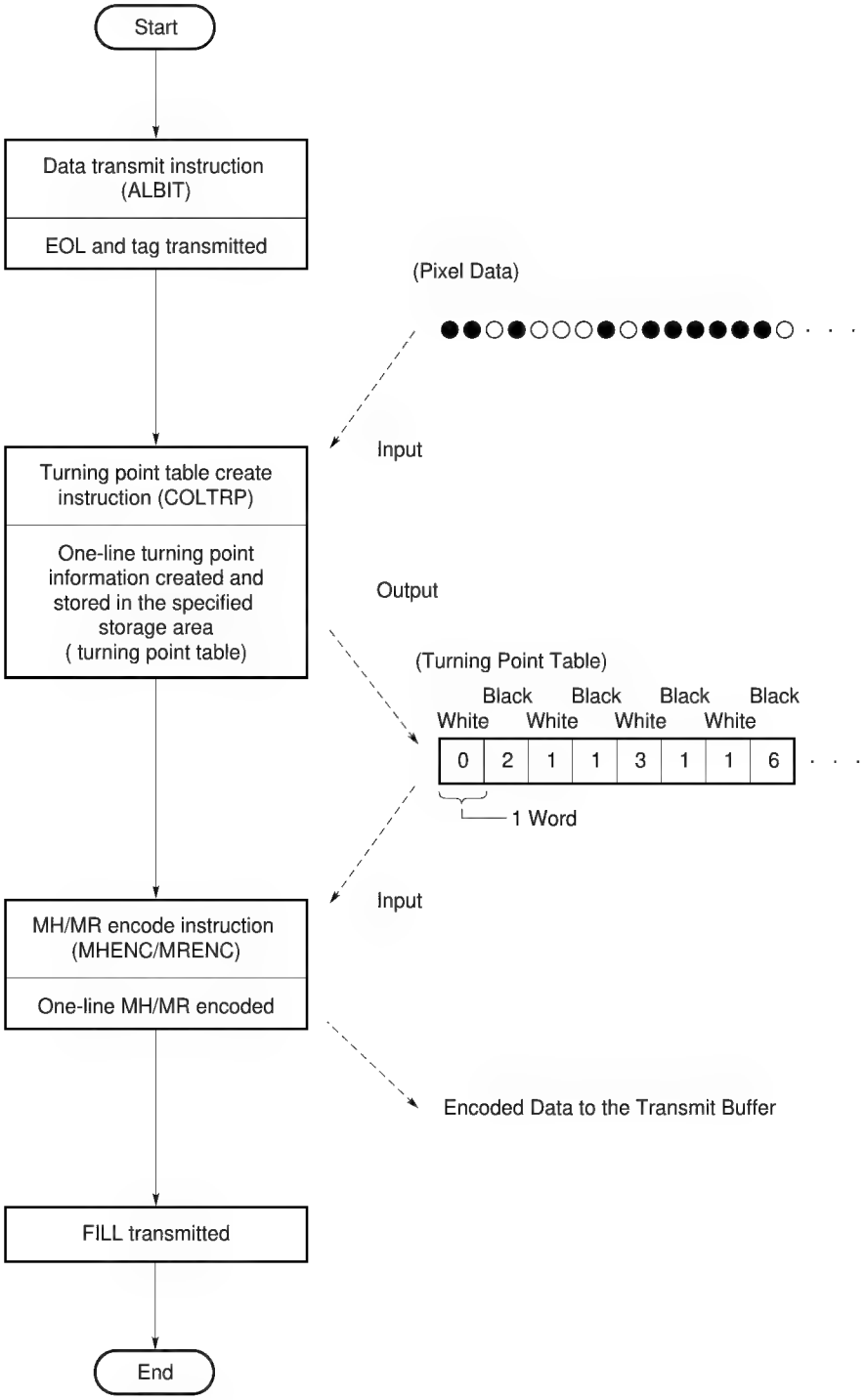
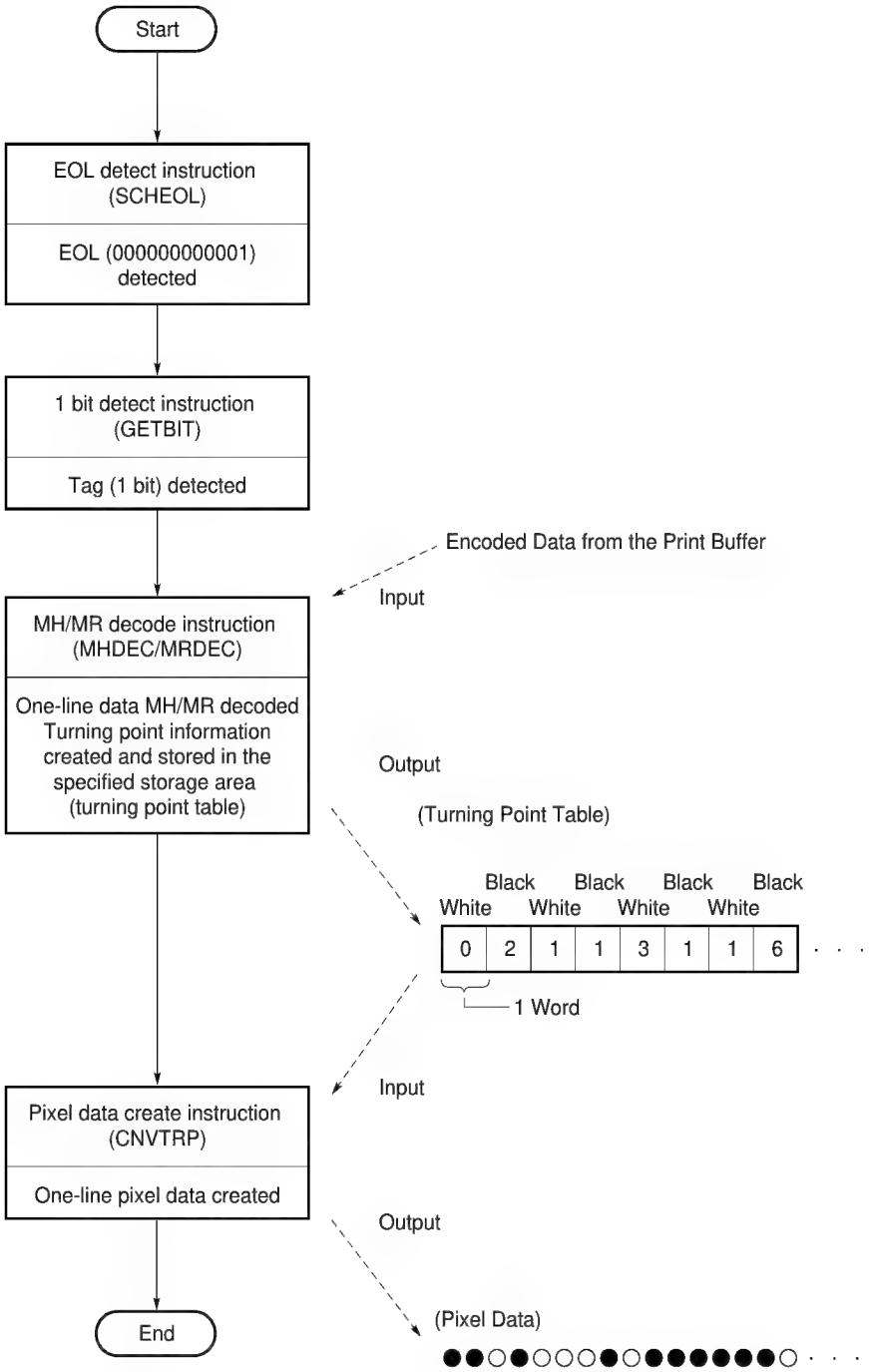


Figure 18-21. One-Line Data Decoding Flow



18.6 Descriptions of Functions of Each Instruction

18.6.1 Data transmit instruction

<Description Format> ALBIT (Add Last BIT)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0

<Number of Bytes> 2

<Number of Clocks>

External Bus Width	Conditions	Number of Clocks
8-bit bus width	CL + CH < 16	14 + n
	CL + CH = 16 and CH = 0	18
	other than above	43–m+n
16-bit bus width	CL + CH < 16	14 + n
	CL + CH = 16 and CH = 0	18
	Other than above	43–m+n

m : CH register set value
n : CL register set value

<Function>
This instruction is intended to output 16 bits or less data such as EOL and FILL to the transmit buffer in encoding. The head of the data is assigned to in the LSB.
When the specified transmit data amounts to 16 bits or more together with the fragmentary code having less than 16 bits undergoing processing, they are output to the transmit buffer. Data above 16 bits or combined data having less than 16 bits is held as parameter in BW (fragmentary code data undergoing processing) and the number of bits of the data is held as parameter in CH (number of fragmentary code bits undergoing processing).
Interrupt request sampling is not carried out.

<Input Parameters>

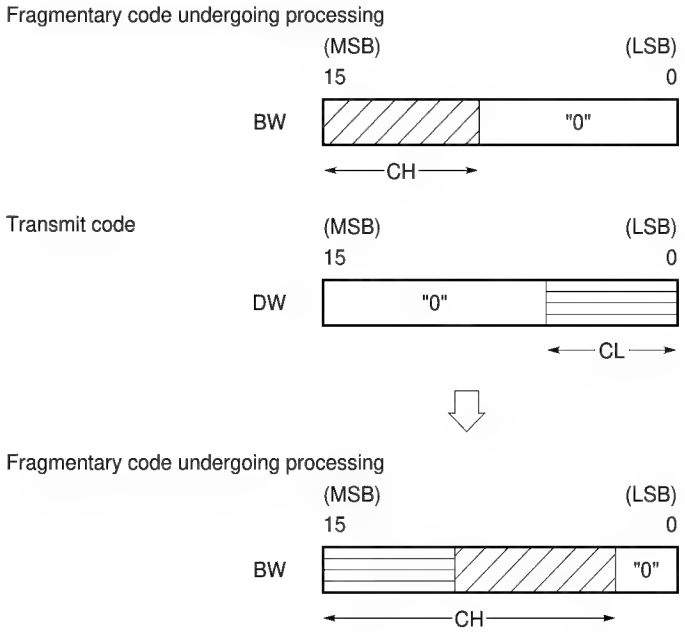
BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
DW : Transmit code data
CL : Number of transmit code data bits
BP : Buffer size (residual byte size of transmit buffer)
DS1 : Transmit buffer segment
IY : Transmit buffer offset

<Output Parameters>

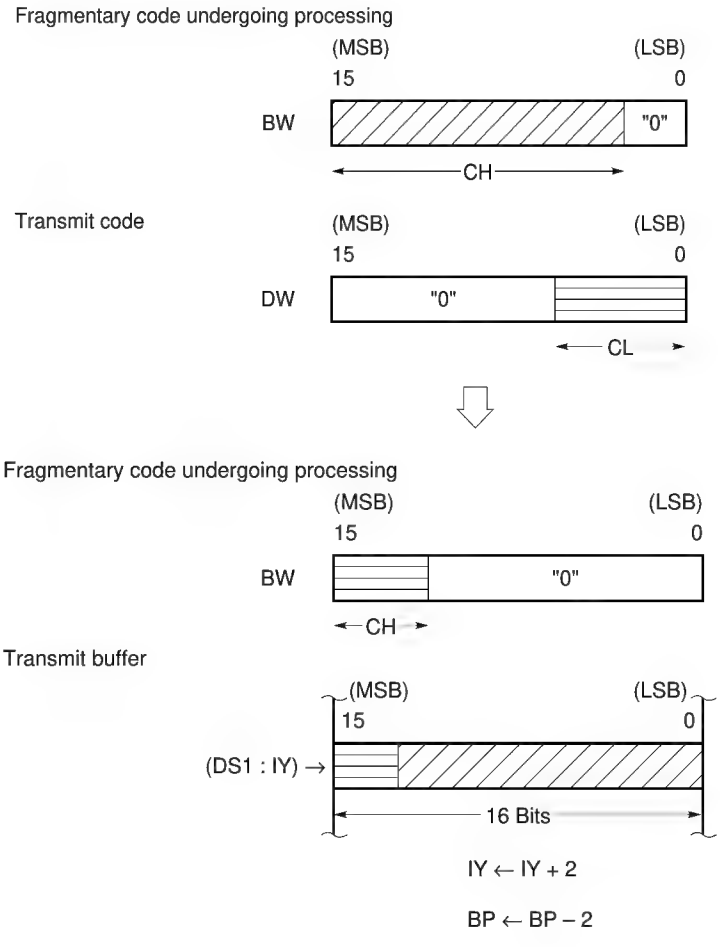
BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
BP : Buffer size (residual byte size of transmit buffer)
DS1 : Transmit buffer segment
IY : Transmit buffer offset

Z Flag	CY Flag	Residual Transmit Buffer	Residual Data
0	Undefined	Existent	–
1	0	None	None
1	1	None	Existent

Example 1. If the fragmentary code undergoing processing and the transmit code do not amount to 16 bits



2. If the fragmentary code undergoing processing and the transmit code amount to 16 bits or more



18.6.2 Turning point table create instruction

<Description Format> COLTRP (Collect Turning Point)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	1

<Number of Bytes> 2

<Number of Clocks>

Indicates the number of clock cycles when there is no break due to an interrupt in execution of the COLTRP instruction.

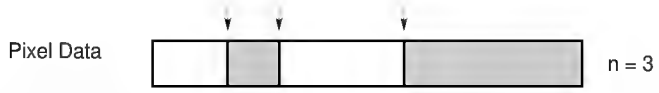
If there is a break due to an interrupt, the number of clock cycles differs depending on the pixel data up to which processing has been completed, and therefore the number is not calculated.

External Bus Width	Conditions	Number of Clocks
8-bit bus width	When pixel data starts from white	17 + 60m + 7n + 2mT
	When pixel data starts from black	24 + 60m + 7n + 2mT
16-bit bus width	When pixel data starts from white	17 + 57m + 7n + mT
	When pixel data starts from black	24 + 57m + 7n + mT

m : Pixel data buffer size (byte)

n : Number of times color of pixel data changes from white to black or from black to white

Example



T : Number of waits in area where pixel data buffer is located.

<Function>

This instruction is intended to generate the turning point information (white/black run length) of one-line pixel data in the specified memory (that is, the turning point table). The head of the data is assigned to in the LSB. Storage in the turning point table starts with the white run-length and continues about each color (white/black) run-length for each word. The last word “FFFFH” is stored as one-line end information in the turning point table.

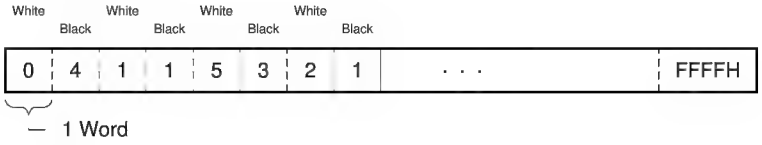
Suspend processing is not carried out. An interrupt request is sampled each time one-word pixel data processing is carried out.

(Pixel data)

●●●●○●○○○○○●●●○○●○…

↓ COLTRP instruction generation

(Turning point table)



<Input Parameters>

AL: Register bank number

00H	(Pixel data buffer segment)	(DS2)	Expansion segment override prefix[Note1]
02H	(Turning point table segment)	(DS3)	
04H	Work area		
06H	Work area		
08H	Pixel data buffer segment	(DS0)	Input data processing information
0AH	Pixel data buffer offset	(pointer)	
0CH	Pixel data buffer size[Note2]	(counter)	
0EH	Turning point table segment	(DS1)	Output data processing information
10H	Turning point table offset	(pointer)	
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	0 clear [Note3]/work area		0 clear required parameters
1EH	0 clear [Note3]/work area		

Notes 1. When DS2: and DS3: are added, 00H nd 02H are referred to instead of 08H and 0EH as the offset values.

2. Byte-wise: Specify using even numbers.

3. Cleared to 0 when a new instruction is processed

Caution The pointer and counter values in the register bank for parameters to be used with each instruction are updated by instruction execution.

18.6.3 MH encode instruction

<Description Format> MHENC (MH Encode)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	1

<Number of Bytes> 2

<Number of Clocks>
Undefined. (Because the number differs depending on the processing pattern).

<Function>
This instruction is intended to input the turning point information of one-line pixel data, execute MH encoding using the specified encoding conversion table and transmit the code to the specified memory (transmit buffer). The head of the data is assigned to in the LSB.
Interrupt request sampling is carried out each time the turning point information (1 word in the turning point table) of each color (white or black) is encoded.
For the MHENC instruction, up to 5183 pixels of the same color (white or black) can be encoded continuously.

<Input Parameters>

AL : Register bank number specification
BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
BP : Buffer size (residual byte size of transmit buffer)
DS1: Transmit buffer segment
IY : Transmit buffer offset

Register banks specified by AL

00H	(Turning point table segment)	(DS2)	Expansion segment override prefix[Note1]
02H	Work area		
04H	Work area		
06H	Work area		
08H	Turning point table segment	(DS0)	Input data processing information
0AH	Encoding line turning point table offset	(pointer)	
0CH	Work area		
0EH	Encoding conversion table segment	(DS1)	Output data processing information
10H	Work area		
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	0 clear [Note2]/work area		0 clear required parameters
1EH	0 clear [Note2]/work area		

Notes 1. When DS2: is added, 00H is referred to instead of 08H as the offset values.

2. Cleared to 0 when a new instruction is processed.
To restart an interrupted process, leave parameters which need to be cleared to 0 as they are.

<Output Parameters>

BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
BP : Buffer size (residual byte size of transmit buffer)
DS1: Transmit buffer segment
IY : Transmit buffer offset

Z Flag	CY Flag	Residual Transmit Buffer	Residual Data	Processing
0	0	Existent	CH value	Terminate
0	1	–	–	–
1	0	None	None	Terminate
1	1	None	Existent	Suspend

★ **Caution If the remaining byte size of the transmit buffer is exactly "0" when the last encoding data is sent to the transmit buffer, the contents of the flags after the MHENC instruction is executed become Z = 1, CY = 1 (No residual transmit buffer, residual data present, processing suspended). Therefore, the MHENC instruction must be re-executed.**

18.6.4 MR encode instruction

<Description Format> MRENC (MR Encode)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1

<Number of Bytes> 2

<Number of Clocks>
Undefined.(because the number differs depending on the processing pattern).

<Function>
This instruction is intended to input the reference line and encoding line turning point information, execute one-line MR encoding and transmit the code to the specified memory (transmit buffer). The head of the data is assigned to in the LSB.
Interrupt request sampling is carried out each time MR encoding is executed for each mode (path mode, vertical mode or horizontal mode).
For MH encoding in the horizontal mode for the MRENC instruction, up to 5183 pixels of the same color (white or black) can be encoded continuously.

<Input Parameters>

AL : Register bank number specification
BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
BP : Buffer size (residual byte size of transmit buffer)
DS1 : Transmit buffer segment
IY : Transmit buffer offset

Register banks specified by AL

00H	(Turning point table segment)	(DS2)	Expansion segment override prefix[Note1]
02H	Work area		
04H	Work area		
06H	Work area		
08H	Turning point table segment	(DS0)	
0AH	Encoding line turning point table offset	(pointer)	
0CH	Reference line turning point table offset	(pointer)	
0EH	Encoding conversion table segment	(DS1)	
10H	Work area		
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	0 clear [Note2]/work area		0-clear-required parameters
1EH	0 clear [Note2]/work area		

Notes 1. When DS2: is added, 00H is referred to instead of 08H as the offset values.

2. Cleared to 0 when a new instruction is processed.
To restart an interrupted process, leave parameters which need to be cleared to 0 as they are.

<Output Parameters>

BW : Fragmentary code data undergoing processing
CH : Number of fragmentary code bits undergoing processing
BP : Buffer size (residual byte size of transmit buffer)
DS1 : Transmit buffer segment
IY : Transmit buffer offset

Z Flag	CY Flag	Residual Transmit Buffer	Residual Data	Processing
0	0	Existent	CH value	Terminate
0	1	–	–	–
1	0	None	None	Terminate
1	1	None	Existent	Suspend

★ **Caution If the remaining byte size of the transmit buffer is exactly "0" when the last encoding data is sent to the transmit buffer, the contents of the flags after the MRENC instruction is executed become Z = 1, CY = 1 (No residual transmit buffer, residual data present, processing suspended). Therefore, the MRENC instruction must be re-executed.**

18.6.5 EOL detect instruction

<Description Format> SCHEOL (Search EOL)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0

<Number of Bytes> 2

<Number of Clocks>

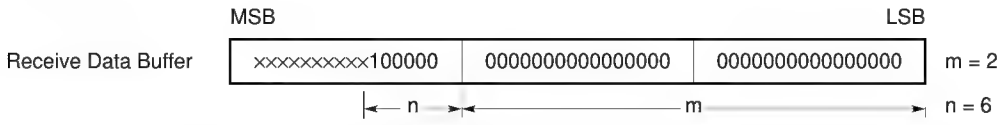
When the conditions in the following table are not met, that is, when EOL is detected after codes other than FILL are detected, the number of clock cycles differs depending on the number of codes other than FILL, and therefore the number is not calculated.

External Bus Width	Conditions	Number of Clocks
8-bit bus width	When EOL is detected at start	24 + 59m + 3n + 2mT
	When EOL with FILL is detected at start	
16-bit bus width	When EOL is detected at start	24 + 56m + 3n + mT
	When EOL with FILL is detected at start	

m : Number of words in which the data is all "0", from the specified position of the receive data buffer until the first "1" is detected

n : Number of consecutive "0" bits of the word in which the first "1" is detected from the specified position of the receive data buffer + 1

Example



T : Number of waits in area where receive data buffer is located

<Function>

This instruction is intended to detect "EOL" (000000000001) in decoding. It detects the first EOL at the specified position in the memory (receive buffer). The head of the data is assigned to in the LSB.

Interrupt request sampling is not executed. To perform interrupt request sampling properly, set the receive buffer size to a small value so that the SCHEOL instruction is interrupted at regular intervals. This prevents the long holds of interrupt requests. The equation for calculating the execution time (maximum) of the SCHEOL instruction is as follows (The refresh cycle, DMA cycle, and bus hold request are not taken into consideration.)

- 16-bit bus width : (214+T) N/16+74+T (clock)
- 8-bit bus width : (110+T) N/8+74+T (clock)

N: Number of bits until EOL detection
T: Number of waits of receive buffer

In detecting the decoded EOL, the following may be the cases:

<1> When all "0" in the specified receive buffer (EOL detection suspension)
<2> When EOL is detected at the beginning
<3> When EOL with FILL is detected at the beginning
<4> When EOL is detected after a code other than FILL is detected

<Input Parameters>

AL : Register bank number

Register banks specified by AL

00H	(Receive buffer segment)	(DS2)	Expansion segment override prefix[Note1]
02H	Work area		
04H	Fragmentary code data undergoing processing		
06H	Number of fragmentary code bits undergoing processing		
08H	Receive buffer segment	(DS0)	
0AH	Receive buffer offset	(pointer)	
0CH	Receive buffer size[Note2]	(counter)	
0EH	Work area		
10H	Work area		
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	Work area		
1EH	0 clear [Note3]/work area		···· 0-clear-required parameters

Notes 1. When DS2: is added, 00H is referred to instead of 08H as the offset values.

2. Byte-wise: Set even numbers other than 0.

3. Cleared to 0 when a new instruction is processed.
To restart an interrupted process, leave parameters which need to be cleared to 0 as they are.

<Output Parameters>

Z Flag	CY Flag	Residual Receive Buffer	End Status	AH Status
0	0	Existent	Normal termination	01, 02
0	1	Existent	End error	03
1	0	None	Normal termination	01, 02
1	1	None	End error	00, 03

The end status can be checked according to numeric values in AH. The relationships between numeric values and end statuses are described below.

AH	End Status
00H	Suspended
01H	EOL detected at the beginning
02H	EOL with FILL detected at the beginning
03H	EOL detected after a code other than EOL has been detected, or EOL detected when receive buffer runs out of empty spaces.

18.6.6 1-bit detect instruction

<Description Format> GETBIT (GET BIT)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	1

<Number of Bytes> 2

<Number of Clocks>

External Bus Width	Conditions	Number of Clocks
8-bit bus width	When number of processing incomplete code bits ≠ 0	13
	When number of processing incomplete code bits = 0	18 + 2T
16-bit bus width	When number of processing incomplete code bits ≠ 0	13
	When number of processing incomplete code bits = 0	15 + T

T : Number of waits in area where receive data buffer is located

<Function>

This instruction is intended to detect 1 bit such as tag in decoding. The head of the data and the incomplete code during the process are assigned to LSB.

This instruction outputs 1 bit from specified memory (receive buffer) and sets it to CY flag.

Interrupt request sampling is not executed.

<Input Parameters>

AL : Register bank number

Register banks specified by AL

00H	(Receive buffer segment)	(DS2)	Expansion segment override prefix[Note1]
02H	Work area		
04H	Fragmentary code data undergoing processing		
06H	Number of fragmentary code bits undergoing processing		
08H	Receive buffer segment	(DS0)	
0AH	Receive buffer offset	(pointer)	
0CH	Receive buffer size[Note2]	(counter)	
0EH	Work area		
10H	Work area		
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	Work area		
1EH	Work area		

Notes 1. When DS2: is added, 00H is referred to instead of 08H as the offset values.

2. Byte-wise: Set even numbers except zero.

<Output Parameters>

CY flag: Detected 1-bit data

Z Flag	Residual Transmit Buffer	
0	Existent	... Receive data buffer size > 0
1	None	... Receive data buffer size = 0

Caution When the receive data buffer size counter (contents of register bank 0CH) is "0", Z flag = 1 irrespective of whether or not any fragmentary code data exists.

18.6.7 MH decode turning point table create instruction

<Description Format> MHDEC (MH Decode)

<Instruction Word Format>

<Number of Bytes> 2

<Number of Clocks>
Undefined. (because the number differs depending on the processing pattern.)

<Function>
This instruction is intended to input the code data of the specified memory (receive buffer), execute MH decoding using the decoding conversion table and transmit the created turning point information to the specified memory (turning point table). The head of the data and the incomplete code during the process are assigned to LSB. That the number of pixel data bits for one line has been reached is recognized and the instruction is terminated by creating the turning point information.
In creating a turning point table from code data for MH decoding, the following may be the cases:

<1> When there is one-line code data in the specified receive buffer and conversion to the turning point information is terminated (normal termination)
<2> When there is code data inproportional to one line in the specified receive buffer and conversion to the turning point information is terminated incompletely (suspension)
<3> When EOL is detected at the beginning
<4> When EOL with FILL is detected at the beginning
<5> When EOL is detected after the data not amounting to the number of pixels for one line
<6> When there is more data to be decoded than the number of pixels for one line
<7> If an abnormal code is detected

Only when the process terminates normally, "FFFFH" (ending codes of one line) is output to the turning point table. Interrupt request sampling is executed each time a decoding operation of creating a turning point in each color (white or black).
For the MHDEC instruction, up to 65535 pixels of one-line can be decoded.
If abnormal codes (continuos 0s) are detected while the MHDEC instruction is executed, the interrupt request is held until "1" is detected next. In this case, set the receive buffer size to a small value so that the MHDEC instruction is interrupted at regular intervals. This prevents the long holds of interrupt requests. The equation for calculating the execution time (maximum) of the MHDEC instruction is as follows (The refresh cycle, DMA cycle, and bus hold request are not taken into consideration.)

- 16-bit bus width: 40 + 2T + (59 + T) N/16 (Clock)
- 8-bit bus width: 44 + 4T + (37 + 2T) N/8 (Clock)

N: Number of bits in which "0" continues
T: Number of waits of turning point table

<Input Parameters>

AL : Register bank number

Register banks specified by AL

00H	(Receive buffer segment)	(DS2)	Expansion segment override prefix[Note1]
02H	(Turning point table segment)	(DS3)	
04H	Fragmentary code data undergoing processing		
06H	Number of fragmentary code bits undergoing processing		
08H	Receive buffer segment	(DS0)	
0AH	Receive buffer offset	(pointer)	
0CH	Receive buffer size[Note2]	(counter)	
0EH	Turning point table segment	(DS1)	
10H	Decoding line turning point table offset		
12H	Total number of pixel bits for one line		
14H	Decoding conversion table segment		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	0 clear [Note3]/work area		0-clear-required parameters
1EH	0 clear [Note3]/work area		

Notes 1. When DS2: and DS3: are added, 00H and 02H are referred to instead of 08H and 0EH as the offset values.

2. Byte-wise: Set even numbers other than 0.

3. Cleared to 0 when a new instruction is processed.
To restart an interrupted process, leave parameters which need to be cleared to 0 as they are.

<Output Parameters>

Z Flag	CY Flag	Residual Transmit Buffer	End Status	AH Status
0	0	Existent	Normal termination	Not changed
0	1	Existent	End error	See **Table 18-4**
1	0	None	Normal termination	Not changed
1	1	None	End error	See **Table 18-4**

The end status can be checked according to numeric values in AH. The relationships between numeric values and end statuses are described below.
When AH = 01H or 02H, there are two end statuses according to the work area value in the offset address 1AH of the register bank specified by the AL register.

Table 18-4. MHDEC Instruction End Status

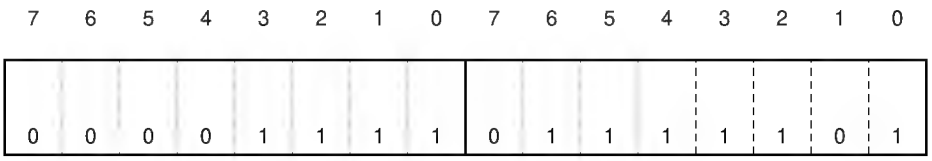
AH	End Status	
00H	Suspended	
01H	Work area = 0	EOL detected at the beginning
	Work area ≠ 0	Make-up code + EOL detected at the beginning
02H	Work area = 0	EOL with FILL detected at the beginning
	Work area ≠ 0	EOL with make-up code + FILL detected at the beginning
03H	EOL detected after the data with the number of pixels not amounting to that for one line	
04H	More data to be decoded exists than the number of pixels for one line.	
05H	Abnormal code detected	

Remark If terminated with abnormal codes detected, the offset of the decoded line turning point table indicates the address of the next word area of the turning point table data which was last output normally.

18.6.8 MR decode turning point table create instruction

<Description Format> MRDEC (MR Decode)

<Instruction Word Format>



<Number of Bytes> 2

<Number of Clocks>
Undefined. (because the number differs depending on the processing pattern)

<Function>
This instruction is intended to input the decoding line code data and reference line turning point information both in the specified memory (receive buffer), execute MR decoding using the decoding conversion table and transmit the created decoding line turning point information to the specified memory (turning point table). The head of the data and the incomplete code during the process are assigned to LSB.
That the number of pixel data bits for one line has been reached is recognized and the instruction is terminated by creating the turning point information.
In creating a turning point table from code data for MR decoding, the following may be the cases:

<1> When there is one-line code data in the specified receive buffer and conversion to the turning point information is terminated (normal termination)
<2> When there is code data inproportional to one line in the specified receive buffer and conversion to the turning point information is terminated incompletely (suspension)
<3> When EOL is detected at the beginning
<4> When EOL with FILL is detected at the beginning
<5> When EOL is detected after the data not amounting to the number of pixels for one line
<6> When there is more data to be decoded than the number of pixels for one line
<7> If an abnormal code is detected

Only when the process terminates normally, "FFFFH" (ending codes of one line) is output to the turning point table.
Interrupt request sampling is carried out each time the decoding is performed in each mode (path mode, vertical mode and horizontal mode).
For the MRDEC instruction, up to 32767 pixels of one-line can be decoded.
If abnormal codes (continuous 0s) are detected while the MRDEC instruction is executed, the interrupt request is held until "1" is detected next. In this case, set the receive buffer size to a small value so that the MRDEC instruction is interrupted at regular intervals. This prevents the long holds of interrupt requests. The equation for calculating the execution time (maximum) of the MRDEC instruction is as follows (The refresh cycle, DMA cycle, and bus hold request are not taken into consideration.)

- 16-bit width: 211 + 6T + (59 + T) N/16 (Clock)
- 8-bit bus width: 227 + 12T + (37 + 2T) N/8 (Clock)

N: Number of bits in which "0" continues
T: Number of waits of turning point table

<Input Parameters>

AL : Register bank number

Register banks specified by AL

00H	(Receive buffer segment)	(DS2)	Expansion segment override prefix[Note1]
02H	(Turning point table segment)	(DS3)	
04H	Fragmentary code data undergoing processing		
06H	Number of fragmentary code bits undergoing processing		
08H	Receive buffer segment	(DS0)	
0AH	Receive buffer offset	(pointer)	
0CH	Receive buffer size[Note2]	(counter)	
0EH	Turning point table segment	(DS1)	
10H	Decoding line turning point table offset		
12H	Reference line turning point table offset		
14H	Decoding conversion table segment		
16H	Work area		
18H	Work area		
1AH	Work area		
1CH	0 clear [Note3]/work area		0-clear-required parameters
1EH	0 clear [Note3]/work area		

Notes 1. When DS2: and DS3: are added, 00H and 02H are referred to instead of 08H and 0EH as the offset values.

2. Byte-wise: Set even numbers other than 0.

3. Cleared to 0 when a new instruction is processed
To restart an interrupted process, leave parameters which need to be cleared to 0 as they are.

<Output Parameters>

Z Flag	CY Flag	Residual Transmit Buffer	End Status	AH Status
0	0	Existent	Normal termination	Not changed
0	1	Existent	End error	See **Table 18-5**
1	0	None	Normal termination	Not changed
1	1	None	End error	See **Table 18-5**

The end status can be checked according to numeric values in AH. The relationships between numeric values and end statuses are described below.

Table 18-5. MRDEC Instruction End Status

AH	End Status
00H	Suspended
01H	EOL detected at the beginning
02H	EOL with FILL detected at the beginning
03H	EOL detected after the data with the number of pixels not amounting to that for one line
04H	More data to be decoded exists than the number of pixels for one line.
05H	Abnormal code detected

Remark If terminated with abnormal codes detected, the offset of the decoded line turning point table indicates the address of the next word area of the turning point table data which was last output normally.

18.6.9 Pixel data create instruction

<Description Format> CNVTRP (Convert Turning Point)

<Instruction Word Format>

7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	0

<Number of Bytes> 2

<Number of Clocks>

Indicates the number of clock cycles when there is no break due to an interrupt in execution of the CNVTRP instruction.

If there is a break due to an interrupt, the number of clock cycles differs depending on the change point information up to which processing to convert it to pixel data has been completed, and therefore the number is not calculated.

External Bus Width	Conditions	Number of Clocks
8-bit bus width	When number (n) of change point informaiton items is even (when change point ends is black)	(34 + 53n)/2 + 14m + 8k + 2 (m + k) T + 2 (n + 1) t
	When number (n) of change point information items is odd (when change point ends in white)	(33 + 53n)/2 + 14m + 8k + 2 (m + k) T + 2 (n + 1) t
16-bit bus width	When number (n) of change point informaiton items is even (when change point ends in black)	(28 + 47n)/2 + 11m + 5k + (m + k) T + (n + 1) t
	When number (n) of change point information items is odd (when change point ends in white)	(27 + 47n)/2 + 11m + 5k + (m + k) T + (n + 1) t

k : Number of times all white/all black pixel data is sent

m : Number of times pixel data other than all white/all black is sent

n : Number of change point information items (sum of number of white change point information items and number of black change point information items)

T : Number of waits in area where pixel data buffer is located

t : Number of waits in area where change point table is located

<Function>

This instruction inputs one-line turning point information of the specified memory (turnign point table), converts it into pixel data, and transmits the resultant data in units of one word (16 bits) to the specified memory (print buffer). The head of the data is assigned to in the LSB.

Interrupt request sampling is carried out each time the turning point information (1 word in the turning point table) of each color (white or black) is converted into pixel data.

If the turning point table indicating pixel data numbers other than × 16 is used, the last pixel data not consisting of 16 bits is not output.

<Input Parameters>

AL : Register bank number

Register banks specified by AL

00H	(Turning point table segment)	(DS2)	Expansion segment override prefix[Note1]
02H	(Print buffer segment)	(DS3)	
04H	Work area		
06H	Work area		
08H	Turning point table segment	(DS0)	
0AH	Turning point table offset	(pointer)	
0CH	Work area		
0EH	Print buffer segment	(DS1)	
10H	Print buffer offset	(pointer)	
12H	Work area		
14H	Work area		
16H	Work area		
18H	Work area		
1AH	Fragmentary pixel data undergoing processing		
1CH	0 clear [Note2]/number of fragmentary pixel bits undergoing processing		0-clear-required parameters
1EH	0 clear [Note2]/work area		

Notes 1. When DS2: and DS3: are aded, 00H and 02H are referred to instead of 08H and 0EH as the offset values.

2. Cleared to 0 when a new instruction is processed

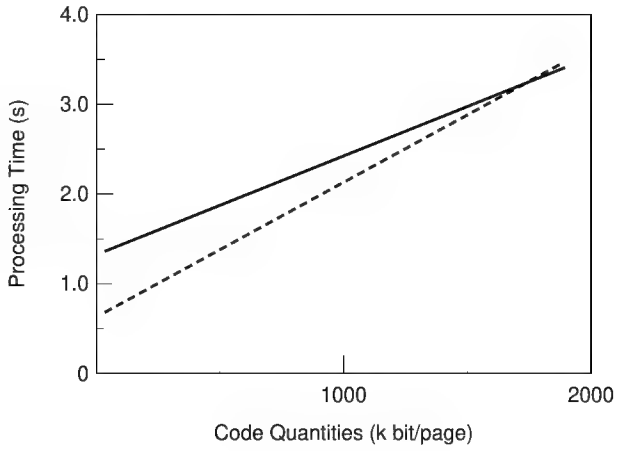
Caution Suspend processing is not carried out.

18.7 Processing Time

The following is the processing time for the amount of code of the corresponding encoding/decoding instruction when the codec instruction is used.

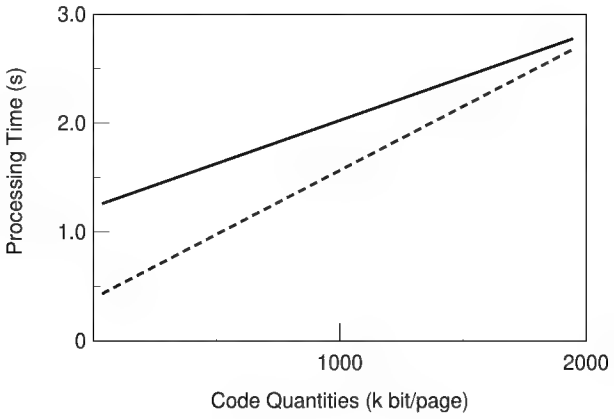
Figure 18-22. Processing Time of MH Encoding/Decoding Instruction

(1) 8-bit bus width, 1 wait



Remarks 1. ——— Encode instruction
--------- Decode instruction
2. Measurement condition: 12.5 MHz.

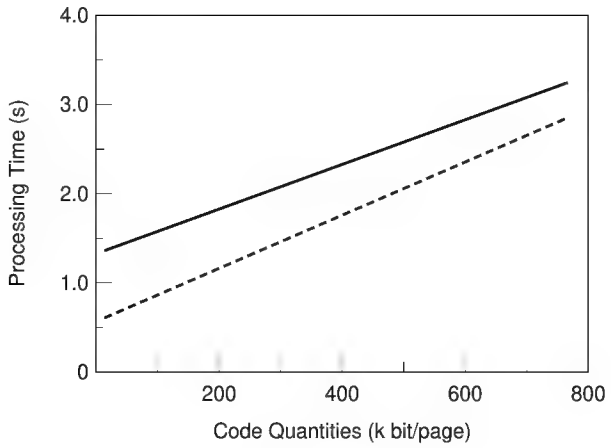
(2) 16-bit bus width, no wait



Remarks 1. ——— Encode instruction
--------- Decode instruction
2. Measurement condition: 12.5 MHz.

Figure 18-23. Processing Time of MR Encoding/Decoding Instruction

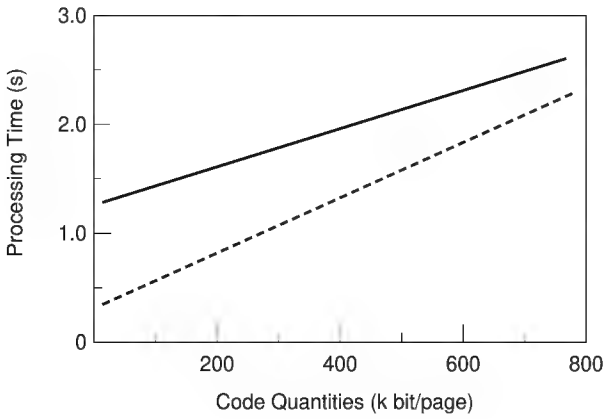
(1) 8-bit bus width, 1 wait



Remarks 1. ——— Encode instruction
--------- Decode instruction

2. Measurement condition: 12.5 MHz, K factor = 4.

(2) 16-bit bus width, no wait



Remarks 1. ——— Encode instruction
--------- Decode instruction

2. Measurement condition: 12.5 MHz, K factor = 4.

[MEMO]

CHAPTER 19 RESET FUNCTION

When low level is input to the $\overline{\text{RESET}}$ input pin, system reset is applied. When a reset is applied, the reset occurs before the operation being executed is completed. The time from when low level is input to the $\overline{\text{RESET}}$ pin until when the state of each pin being reset is 1 μs Max. The hardware related registers enter the state shown in Table 19-1. When the $\overline{\text{RESET}}$ input becomes high, the reset state is reset and program execution begins. Initialize the contents of the registers in the program, as required. To fully display the performance of V55PI, the PRC register, RFM register, MBC register, and PWC register must be set to values corresponding to the respective applications at the least.

Table 19-1. Hardware States After Reset (1/4)

Hardware (Symbol)			Address	State After Reset
Program counter		PC	–	0000H
Program status word		PSW	–	F002H
On-chip RAM	Data memory		–	Undefined
	General-purpose register	AW, CW, DW, BW SP, BP, IX, IY	–	Undefined
	Segment register	SS, DS0 to DS3	–	0000H
		PS	–	FFFFH
Port	Port register	P0 to P8	FFF00H to FFF08H	Undefined
	Port mode register	PM0 PM2 to PM5 PM7, PM8	FFF10H FFF12H to FFF15H FFF17H, FFF18H	FFH
	Port mode control register	PMC2 to PMC5 PMC7, PMC8	FFF22H to FFF25H FFF27H, FFF28H	00H
	Port read control register	PRDC	FFF0CH	00H
Real-time output port	Real-time output port control register	RTPC	FFF2CH	40H
	Real-time output port delay specification register	RTPD	FFF2DH	Undefined
	Port 7 buffer	P7H, P7L	FFF2EH to FFF2FH	Undefined
	Real-time output port	RTP	FFF0EH	Undefined
Timer unit	Timer register	TM0 to TM3	FFF40H to FFF47H	00H
	Timer capture register	CT00, CT01 CT10	FFF48H to FFF4BH FFF50H to FFF51H	Undefined
	Timer compare register	CM00, CM01 CM10, CM11 CM20, CM21 CM22, CM23 CM30, CM31	FFF4CH to FFF4FH FFF52H to FFF55H FFF58H to FFF5BH FFF5CH to FFF5FH FFF64H to FFF67H	Undefined

Table 19-1. Hardware States After Reset (2/4)

Hardware (Symbol)				Address	State After Reset
Timer unit	Timer control register	TMC	TMC0	FFF30H	000H
			TMC1	FFF31H	
	Timer output control register	TOC	TOC0	FFF32H	000H
			TOC1	FFF33H	
Software interval timer	Software timer counter	STC		FFFE0H to FFFE1H	Undefined
	Software timer counter compare register	STMC		FFFE2H to FFFE3H	FFFFH
PWM unit	PWM register	PWM		FFF6CH	Undefined
	PWM control register	PWMC		FFF6DH	00H
DMA controller	DMA mode register	DMAM0, DMAM1		FFF9CH, FFFBCH	E0H
	DMA control register	DMAC0, DMAC1		FFF9DH, FFFBDH	00H
	Terminal counter	TC0	TC0L	FFF80H, FFF81H	Undefined
			TC0H	FFF82H	
		TC1	TC1L	FFFA0H, FFFA1H	
			TC1H	FFFA2H	
	Terminal counter modulo register	TCM0	TCM0L	FFF84H, FFF85H	Undefined
			TCM0H	FFF86H	
		TCM1	TCM1L	FFFA4H, FFFA5H	
			TCM1H	FFFA6H	
	DMA memory address register	MAR0	MAR0L	FFF90H, FFF91H	Undefined
			MAR0H	FFF92H	
		MAR1	MAR1L	FFFB0H, FFFB1H	
			MAR1H	FFFB2H	
	DMA up/down counter	UDC0	UDC0L	FFF88H, FFF89H	Undefined
			UDC0H	FFF8AH	
		UDC1	UDC1L	FFFA8H, FFFA9H	
			UDC1H	FFFAAH	
	DMA compare register	DCM0	DCM0L	FFF8CH, FFF8DH	Undefined
			DCM0H	FFF8EH	
		DCM1	DCM1L	FFFACH, FFFADH	
			DCM1H	FFFAEH	
	DMA read/write pointer	DPTC0	DPTC0L	FFF94H, FFF95H	Undefined
			DPTC0H	FFF96H	
		DPTC1	DPTC1L	FFFB4H, FFFB5H	
			DPTC1H	FFFB6H	
	DMA status register	DMAS		FFF9EH	00H

Table 19-1. Hardware States After Reset (3/4)

Hardware (Symbol)				Address	State After Reset
A/D converter	A/D converter mode register	ADM		FFE20H	00H
	A/D conversion result register	ADCR0		FFE00H	Undefined
		ADCR1		FFE02H	
		ADCR2		FFE04H	
		ADCR3		FFE06H	
Parallel interface	Parallel interface buffer	PAB		FFE10H	Undefined
	Parallel interface control register	PAC0		FFE18H	90H
		PAC1		FFE19H	03H
	Parallel interface status register	PAS		FFE1AH	40H
	Parallel interface acknowledge interval register	PAI1		FFE1CH	Undefined
		PAI2		FFE1DH	
UART/CSI	Protocol select register	ASP		FFF7FH	00H
	UART mode register/ CSI mode register	UARTM0/CSIM0		FFF73H	00H
		UARTM1/CSIM1		FFF7BH	
	UART status register/ SBI control register	UARTS0/SBIC0		FFF74H	20H/00H
		UARTS1		FFF7CH	20H
	Receive buffer	RxB0		FFF76H	Undefined
		RxB1		FFF7EH	
	UART transmit buffer/ CSI shift register	TxB0/SIO0		FFF75H	Undefined
		TxB1/SIO1		FFF7DH	
	Receive baud rate generator register	RxBRG0		FFF71H	Undefined
		RxBRG1		FFF79H	
	Transmit baud rate generator register	TxBRG0		FFF70H	Undefined
		TxBRG1		FFF78H	
	Prescaler register	PRS0		FFF72H	00H
		PRS1		FFF7AH	
Interrupt	Interrupt mode control register	IMC		FFEC5H	80H
	Interrupt mask flag register	MK0	MK0L	FFEC0H	FFH
			MK0H	FFEC1H	
		MK1	MK1L	FFEC2H	
			MK1H	FFEC3H	
	Interrupt request control register	IC09 to IC14		FFEC9H to FFECEH	43H
		IC16 to IC32		FFED0H to FFEE1H	
		IC36 to IC37		FFEE4H to FFEE5H	
	In-service priority register	ISPR		FFEC4H	00H
	External interrupt mode register	INTM	INTM0	FFF34H	00H
			INTM1	FFF35H	

Table 19-1. Hardware States After Reset (4/4)

Hardware (Symbol)		Address	State After Reset
Standby control register	STBC	FFFEEH	Undefined[Note]
Processor control register	PRC	FFFEFH	EEH
Programmable wait control register	PWC0	FFFE8H	EAH
	PWC1	FFFE9H	AAH
Refresh mode register	RFM	FFFECH	77H
Memory block control register	MBC	FFFEAH	FCH
Watchdog timer mode register	WDM	FFF60H	00H

Note At power-on reset: 00H, Otherwise: No change

APPENDIX A LIST OF MNEMONIC

A mnemonic list is shown in Table A-1.

Table A-1. List of Mnemonics (1/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Data transfer instructions	MOV	reg, reg'	1	0	0	0	1	0	1	W	1	1	reg			reg'			2	reg ← reg'						
		mem, reg	1	0	0	0	1	0	0	W	mod		reg			mem			2-4	(mem) ← reg						
		reg, mem	1	0	0	0	1	0	1	W	mod		reg			mem			2-4	reg ← (mem)						
		mem, imm	1	1	0	0	0	1	1	W	mod	0	0	0		men			3-6	(mem) ← imm						
		reg, imm	1	0	1	1	W		reg										2-3	reg ← imm						
		acc, dmem	1	0	1	0	0	0	0	W									3	If W = 0 : AL ← (dmem) If W = 1 : AH ← (dmem + 1), AL ← (dmem)						
		dmem, acc	1	0	1	0	0	0	1	W									3	If W = 0 : (dmem) ← AL If W = 1 : (dmem + 1) ← AH, (dmem) ← AL						
		sreg, reg16	1	0	0	0	1	1	1	0	1	1	0	sreg		reg			2	sreg ← reg16 sreg : SS, DS0, DS1						
		sreg, mem16	1	0	0	0	1	1	1	0	mod	0	sreg			mem			2-4	sreg ← (mem16) sreg : SS, DS0, DS1						
		reg16, sreg	1	0	0	0	1	1	0	0	1	1	0	sreg		reg			2	reg16 ← sreg						
		mem16, sreg	1	0	0	0	1	1	0	0	mod	0	sreg			mem			2-4	(mem16) ← sreg						
		DS0, reg16, mem32	1	1	0	0	0	1	0	1	mod		reg			mem			2-4	reg16 ← (mem32) DS0 ← (mem32 + 2)						
		DS1, reg16, mem32	1	1	0	0	0	1	0	0	mod		reg			mem			2-4	reg16 ← (mem32) DS1 ← (mem32 + 2)						
		AH, PSW	1	0	0	1	1	1	1	1									1	AH ← S, Z, 0, AC, 0, P, $\overline{\text{IBRK}}$, CY						
		PSW, AH	1	0	0	1	1	1	1	0									1	S, Z, 0, AC, 0, P, $\overline{\text{IBRK}}$, CY ← AH	×	×		×	×	×

Table A-1. List of Mnemonics (2/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Data transfer instructions	LDEA	reg16, mem16	1	0	0	0	1	1	0	1	mod reg mem		2-4	reg16 ← mem16						
	TRANS TRANSB[Note1]	(src-table)	1	1	0	1	0	1	1	1			1	AL ← (BW + AL)						
	XCH	reg, reg'	1	0	0	0	0	1	1	W	1 1 reg reg'		2	reg ↔ reg'						
		mem, reg reg, mem	1	0	0	0	0	1	1	W	mod reg mem		2-4	(mem) ↔ reg						
		AW, reg16 reg16, AW	1	0	0	1	0	reg					1	AW ↔ reg16						
	MOVSPA[Note2]		0	0	0	0	1	1	1	1	0 0 1 0 0 1 0 1		2	New register bank SS, SP ← old register bank SS, SP						
	MOVSPB[Note2]	reg16	0	0	0	0	1	1	1	1	1 0 0 1 0 1 0 1		3	SS, SP of register bank indicated by reg16 ← present register bank SS, SP						
			1	1	1	1	1	reg												

Notes 1. The operand can be omitted in the case of the TRANS instruction. The TRANSB instruction has no operand.

2. This instruction is not provided in the V20 or V30.

Table A-1. List of Mnemonics (3/33)

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Repeat prefixes	REPC		0 1 1 0 0 1 0 1	1	While CW ≠ 0, the following byte primitive block transfer instruction is executed and CW is decremented (–1). If there is a pending interrupt, it is serviced. If CY ≠ 1 the loop is exited.						
	REPNC		0 1 1 0 0 1 0 0	1	Same as above If CY ≠ 0 the loop is exited.						
	REP REPE REPZ		1 1 1 1 0 0 1 1	1	While CW ≠ 0, the following byte primitive block transfer instruction is executed and CW is decremented (–1). If there is a pending interrupt, it is serviced. If the primitive block transfer instruction is CMPBK or CMPM and Z ≠ 1 the loop is exited.						
	REPNE REPNZ		1 1 1 1 0 0 1 0	1	Same as above If Z ≠ 0 the loop is exited.						
Primitive block transfer instructions	MOVBK MOVBKB MOVBKW	dst-block, src-block	1 0 1 0 0 1 0 W	1	If W = 0: (IY) ← (IX) DIR = 0 IX ← IX + 1, IY ← IY + 1 DIR = 1 IX ← IX – 1, IY ← IY – 1 If W = 1: (IY + 1, IY) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX – 2, IY ← IY – 2						
	CMPBK CMPBKB CMPBKW	src-block, dst-block	1 0 1 0 0 1 1 W	1	If W = 0: (IX) – (IY) DIR = 0: IX ← IX + 1, IY ← IY + 1 DIR = 1: IX ← IX – 1, IY ← IY – 1 If W = 1: (IX + 1, IX) – (IY + 1, IY) DIR = 0: IX ← IX + 2, IY ← IY + 2 DIR = 1: IX ← IX – 2, IY ← IY – 2	×	×	×	×	×	×

Table A-1. List of Mnemonics (4/33)

Instruction Group	Mnemonic	Operands	Operation Code																Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Primitive block transfer instructions	CMPM CMPMB CMPMW	dst-block	1	0	1	0	1	1	1	W									1	If W = 0: AL – (IY) DIR = 0: IY ← IY + 1, DIR = 1: IY ← IY – 1 If W = 1: AW – (IY + 1, IY) DIR = 0: IY ← IY + 2, DIR = 1: IY ← IY – 2	×	×	×	×	×	×
	LDM LDMB LDMW	src-block	1	0	1	0	1	1	0	W									1	If W = 0: AL ← (IX) DIR = 0: IX ← IX + 1, DIR = 1: IX ← IX – 1 If W = 1: AW ← (IX + 1, IX) DIR = 0: IX ← IX + 2, DIR = 1: IX ← IX – 2						
	STM STMB STMW	dst-block	1	0	1	0	1	0	1	W									1	If W = 0: (IY) ← AL DIR = 0: IY ← IY + 1, DIR = 1: IY ← IY – 1 If W = 1: (IY + 1, IY) ← AW DIR = 0: IY ← IY + 2, DIR = 1: IY ← IY – 2						
Bit field manipulation instructions	INS	reg8, reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	0	1	3	16-bit field ← AW						
			1	1	reg'			reg																		
		reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	0	1	4	16-bit field ← AW						
			1	1	0	0	0	reg																		
	EXT	reg8, reg8'	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	3	AW ← 16-bit field						
			1	1	reg'			reg																		
		reg8, imm4	0	0	0	0	1	1	1	1	0	0	1	1	1	0	1	1	4	AW ← 16-bit field						
			1	1	0	0	0	reg'																		

Table A-1. List of Mnemonics (5/33)

Instruction Group	Mnemonic	Operands	Operation Code																Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Input/output instructions	IN[Note]	acc, imm8	1	1	1	0	0	1	0	W									2	If W = 0: AL ← (imm8) If W = 1: AH ← (imm8 + 1), AL ← (imm8)						
		acc, DW	1	1	1	0	1	1	0	W									1	If W = 0: AL ← (DW) If W = 1: AH ← (DW + 1), AL ← (DW)						
	OUT[Note]	imm8, acc	1	1	1	0	0	1	1	W									2	If W = 0: (imm8) ← AL If W = 1: (imm8 + 1) ← AH, (imm8) ← AL						
		DW, acc	1	1	1	0	1	1	1	W									1	If W = 0: (DW) ← AL If W = 1: (DW + 1) ← AH, (DW) ← AL						
Primitive input/outut instructions	INM[Note]	dst-block, DW	0	1	1	0	1	1	0	W									1	If W = 0: (IY) ← (DW) DIR = 0: IY ← IY + 1, DIR = 1: IY ← IY – 1 If W = 1: (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2, DIR = 1: IY ← IY – 2						
	OUTM[Note]	DW, src-block	0	1	1	0	1	1	1	W									1	If W = 0 (DW) ← (IX) DIR = 0 IX ← IX + 1, DIR = 1: IX ← IX – 1 If W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2, DIR = 1: IX ← IX –2						

Note If $\overline{IBRK}$ = 0, a software interrupt is generated automatically and the instruction is not executed.

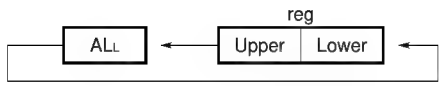
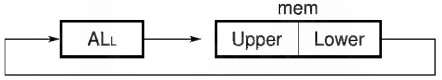
Table A-1. List of Mnemonics (6/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Addition/ subtraction instructions	ADD	reg, reg'	0	0	0	0	0	0	1	W	1 1 reg reg'		2	reg ← reg + reg'	×	×	×	×	×	×
		mem, reg	0	0	0	0	0	0	0	W	mod reg mem		2-4	(mem) ← (mem) + reg	×	×	×	×	×	×
		reg, mem	0	0	0	0	0	0	1	W	mod reg mem		2-4	reg ← reg + (mem)	×	×	×	×	×	×
		reg, imm	1	0	0	0	0	0	S	W	1 1 0 0 0 reg		3-4	reg ← reg + imm	×	×	×	×	×	×
		mem, imm	1	0	0	0	0	0	S	W	mod 0 0 0 mem		3-6	(mem) ← (mem) + imm	×	×	×	×	×	×
		acc, imm	0	0	0	0	0	1	0	W			2-3	If W = 0 : AL ← AL + imm	×	×	×	×	×	×
	ADDC	reg, reg'	0	0	0	1	0	0	1	W	1 1 reg reg'		2	reg ← reg + reg' + CY	×	×	×	×	×	×
		mem, reg	0	0	0	1	0	0	0	W	mod reg mem		2-4	(mem) ← (mem) + reg + CY	×	×	×	×	×	×
		reg, mem	0	0	0	1	0	0	1	W	mod reg mem		2-4	reg ← reg + (mem) + CY	×	×	×	×	×	×
		reg, imm	1	0	0	0	0	0	S	W	1 1 0 1 0 reg		3-4	reg ← reg + imm + CY	×	×	×	×	×	×
		mem, imm	1	0	0	0	0	0	S	W	mod 0 1 0 mem		3-6	(mem) ← (mem) + imm + CY	×	×	×	×	×	×
		acc, imm	0	0	0	1	0	1	0	W			2-3	If W = 0 : AL ← AL + imm + CY If W = 1 : AW ← AW + imm + CY	×	×	×	×	×	×
	SUB	reg, reg'	0	0	1	0	1	0	1	W	1 1 reg reg'		2	reg ← reg – reg'	×	×	×	×	×	×
		mem, reg	0	0	1	0	1	0	0	W	mod reg mem		2-4	(mem) ← (mem) – reg	×	×	×	×	×	×
		reg, mem	0	0	1	0	1	0	1	W	mod reg mem		2-4	reg ← reg – (mem)	×	×	×	×	×	×
		reg, imm	1	0	0	0	0	0	S	W	1 1 1 0 1 reg		3-4	reg ← reg – imm	×	×	×	×	×	×
		mem, imm	1	0	0	0	0	0	S	W	mod 1 0 1 mem		3-6	(mem) ← (mem) – imm	×	×	×	×	×	×
		acc, imm	0	0	1	0	1	1	0	W			2-3	If W = 0 : AL ← AL – imm If W = 1 : AW ← AW – imm	×	×	×	×	×	×

Table A-1. List of Mnemonics (7/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Addition/ subtraction instructions	SUBC	reg, reg'	0	0	0	1	1	0	1	W	1	1	reg				reg'		2	reg ← reg – reg' – CY	×	×	×	×	×	×
		mem, reg	0	0	0	1	1	0	0	W	mod		reg			mem			2-4	(mem) ← (mem) – reg – CY	×	×	×	×	×	×
		reg, mem	0	0	0	1	1	0	1	W	mod		reg			mem			2-4	reg ← reg – (mem) – CY	×	×	×	×	×	×
		reg, imm	1	0	0	0	0	0	S	W	1	1	0	1	1	reg			3-4	reg ← reg – imm – CY	×	×	×	×	×	×
		mem, imm	1	0	0	0	0	0	S	W	mod		0	1	1	mem			3-6	(mem) ← (mem) – imm – CY	×	×	×	×	×	×
		acc, imm	0	0	0	1	1	1	0	W									2-3	If W = 0 : AL ← AL – imm – CY If W = 1 : AW ← AW – imm – CY	×	×	×	×	×	×

Table A-1. List of Mnemonics (8/33)

Instruction Group	Mnemonic	Operands	Operation Code															Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
BCD operation instruction	ADD4S[Note1]	(dst-string, src-string)	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	2	dst BCD string ← dst BCD string + src BCD string[Note2]	U	×	U	U	U	×
	SUB4S[Note1]	(dst-string, src-string)	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	2	dst BCD string ← dst BCD string – src BCD string[Note2]	U	×	U	U	U	×
	CMP4S[Note1]	(dst-string, src-string)	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	2	dst BCD string – src BCD string[Note2]	U	×	U	U	U	×
	ROL4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	3							
			1	1	0	0	0	reg																		
		mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	3-5							
			mod		0	0	0	mem																		
	ROR4	reg8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	3							
			1	1	0	0	0	reg																		
		mem8	0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	3-5							
			mod		0	0	0	mem																		

Remark n is 1/2 the number of BCD digits.

Notes 1. The operand can be omitted.

2. The number of BCD digits is given by the CL register: a value between 1 and 254 can be set.

Table A-1. List of Mnemonics (9/33)

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Increment/ decrement instructions	INC	reg8	1 1 1 1 1 1 1 0 / 1 1 0 0 0 reg	2	reg8 ← reg8 + 1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W / mod 0 0 0 mem	2-4	(mem) ← (mem) + 1	×		×	×	×	×
		reg16	0 1 0 0 0 reg	1	reg16 ← reg16 + 1	×		×	×	×	×
	DEC	reg8	1 1 1 1 1 1 1 0 / 1 1 0 0 1 reg	2	reg8 ← reg8 – 1	×		×	×	×	×
		mem	1 1 1 1 1 1 1 W / mod 0 0 1 mem	2-4	(mem) ← (mem) – 1	×		×	×	×	×
		reg16	0 1 0 0 1 reg	1	reg16 ← reg16 – 1	×		×	×	×	×
Multiplication instructions	MULU	reg8	1 1 1 1 0 1 1 0 / 1 1 1 0 0 reg	2	AW ← AL × reg8 AH = 0 : CY ← 0, V ← 0 AH ≠ 0 : CY ← 1, V ← 1	U	×	×	U	U	U
		mem8	1 1 1 1 0 1 1 0 / mod 1 0 0 mem	2-4	AW ← AL × (mem8) AH = 0 : CY ← 0, V ← 0 AH ≠ 0 : CY ← 1, V ← 1	U	×	×	U	U	U
		reg16	1 1 1 1 0 1 1 1 / 1 1 1 0 0 reg	2	DW, AW ← AW × reg16 DW = 0 : CY ← 0, V ← 0 DW = 1 : CY ← 1, V ← 1	U	×	×	U	U	U
		mem16	1 1 1 1 0 1 1 1 / mod 1 0 0 mem	2-4	DW, AW ← AW × (mem16) DW = 0 : CY ← 0, V ← 0 DW = 1 : CY ← 1, V ← 1	U	×	×	U	U	U

Table A-1. List of Mnemonics (10/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Multiplication instructions	MUL	reg8	1	1	1	1	0	1	1	0	1	1	1	0	1	reg			2	AW ← AL × reg8 AH = AL sign extention: CY ← 0, V ← 0 AH ≠ AL sign extention: CY ← 1, V ← 1	U	×	×	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod		1	0	1	mem			2-4	AW ← AL × (mem8) AH = AL sign extention: CY ← 0, V ← 0 AH ≠ AL sign extention: CY ← 1, V ← 1	U	×	×	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	0	1	reg			2	DW, AW ← AW × reg16 DW = AW sign extention: CY ← 0, V ← 0 DW ≠ AW sign extention: CY ← 1, V ← 1	U	×	×	U	U	U
		mem16	1	1	1	1	0	1	1	1	mod		1	0	1	mem			2-4	DW, AW ← AW × (mem16) DW = AW sign extention: CY ← 0, V ← 0 DW ≠ AW sign extention: CY ← 1, V ← 1	U	×	×	U	U	U
		reg16, (reg16,)[Note] imm8	0	1	1	0	1	0	1	1	1	1	reg			reg			3	reg16 ← reg16 × imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	U	×	×	U	U	U
		reg16, mem16, imm8	0	1	1	0	1	0	1	1	mod		reg			mem			3-5	reg16 ← (mem16) × imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	U	×	×	U	U	U
		reg16, (reg16,)[Note] imm16	0	1	1	0	1	0	0	1	1	1	reg			reg			4	reg16 ← reg16 × imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	U	×	×	U	U	U
		reg16, mem16, imm16	0	1	1	0	1	0	0	1	mod		reg			mem			4-6	reg16 ← (mem16) × imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	U	×	×	U	U	U

Note The 2nd operand can be omitted, in which case the same register as the 1st operand is taken as being specified.

Table A-1. List of Mnemonics (11/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Unsigned division instructions	DIVU	reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg			2	temp ← AW If temp / reg8 ≤ FFH AH← temp%reg8, AL ← temp / reg8 If temp / reg8 > FFH (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod		1	1	0	mem			2-4	temp ← AW If temp / (mem8) ≤ FFH AH← temp%(mem8), AL ← temp / (mem8) If temp / (mem8) > FFH (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg			2	temp ← DW, AW If temp / reg16 ≤ FFFFH DW ← temp%reg16, AW ← temp / reg16 If temp / reg16 > FFFFH (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
		mem16	1	1	1	1	0	1	1	1	mod		1	1	0	mem			2-4	temp ← DW, AW If temp / (mem16) ≤ FFFFH DW ← temp%(mem16), AW ← temp / (mem16) If temp / (mem16) > FFFFH (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP – 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U

Table A-1. List of Mnemonics (12/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Signed division instructions	DIV	reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg			2	temp ← AW If temp / reg8 > 0 and temp / reg8 ≤ 7FH or temp / reg8 < 0 and temp / reg8 > 0 – 7FH – 1 AH ← temp%reg8, AL ← temp / reg8 If temp / reg8 > 0 and temp / reg8 > 7FH or temp / reg8 > 0 and temp / reg8 < 0 – 7FH – 1 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
		mem8	1	1	1	1	0	1	1	0	mod	1	1	1	mem				2-4	temp ← AW If temp / (mem8) > 0 and temp / (mem8) ≤ 7FH or temp / (mem8) < 0 and temp / (mem8) > 0 – 7FH – 1 AH ← temp%(mem8), AL ← temp / (mem8) If temp / (mem8) > 0 and temp / (mem8) > 7FH or temp / (mem8) > 0 and temp / (mem8) < 0 – 7FH – 1 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
		reg16	1	1	1	1	0	1	1	1	1	1	1	1	1	reg			2	temp ← DW, AW If temp / reg16 > 0 and temp / reg16 ≤ 7FFFH or temp / reg16 < 0 and temp / reg16 > 0 – 7FFFH – 1 DW ← temp%reg16, AW ← temp / reg16 If temp / reg16 > 0 and temp / reg16 > 7FFFH or temp / reg16 > 0 and temp / reg16 < 0 – 7FFFH – 1 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U

Table A-1. List of Mnemonics (13/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Signed division instructions	DIV	mem16	1	1	1	1	0	1	1	1	mod 1 1 1 mem	2-4	temp ← DW, AW If temp / (mem16) > 0 and temp / (mem16) ≤ 7FFFH or temp / (mem16) < 0 and temp / (mem16) > 0 – 7FFFH – 1 DW ← temp%(mem16), AW ← temp / (mem16) If temp / (mem16) > 0 and temp / (mem16) > 7FFFH or temp / (mem16) > 0 and temp / (mem16) < 0 – 7FFFH – 1 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0)	U	U	U	U	U	U
BCD adjustment instructions	ADJBA		0	0	1	1	0	1	1	1		1	If AL ∧0FH > 9 or AC = 1: AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL∧ 0FH	×	×	U	U	U	U
	ADJ4A		0	0	1	0	0	1	1	1		1	If AL ∧0FH > 9 or AC = 1: AL ← AL + 6, CY ← CY∨ AC, AC ← 1 If AL > 9FH or CY = 1: AL ← AL + 60H, CY ← 1	×	×	U	×	×	×
	ADJBS		0	0	1	1	1	1	1	1		1	If AL ∧0FH > 9 or AC = 1: AL ← AL – 6, AH ← AH – 1, AC ← 1, CY ← AC, AL ← AL∧ 0FH	×	×	U	U	U	U
	ADJ4S		0	0	1	0	1	1	1	1		1	If AL ∧0FH > 9 or AC = 1: AL ← AL – 6, CY ← CY∨ AC, AC ← 1 If AL > 9FH or CY = 1: AL ← AL – 60H, CY ← 1	×	×	U	×	×	×
Data conversion instructions	CVTBD		1	1	0	1	0	1	0	0	0 0 0 0 1 0 1 0	2	AH ← AL ÷ 0AH, AL ← AL%0AH	U	U	U	×	×	×
	CVTDB		1	1	0	1	0	1	0	1	0 0 0 0 1 0 1 0	2	AH ← 0, AL ← AH × 0AH + AL	U	U	U	×	×	×
	CVTBW		1	0	0	1	1	0	0	0		1	If AL < 80H: AH ← 0, Otherwise: AH ← FFH						
	CVTWL		1	0	0	1	1	0	0	1		1	If AW < 8000H: DW ← 0, Otherwise: DW ← FFFFH						

Table A-1. List of Mnemonics (14/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Comparison instructions	CMP	reg, reg'	0	0	1	1	1	0	1	W	1	1	reg				reg'		2	reg – reg'	×	×	×	×	×	×
		mem, reg	0	0	1	1	1	0	0	W	mod		reg			mem			2-4	(mem) – reg	×	×	×	×	×	×
		reg, mem	0	0	1	1	1	0	1	W	mod		reg			mem			2-4	reg – (mem)	×	×	×	×	×	×
		reg, imm	1	0	0	0	0	0	S	W	1	1	1	1	1	reg			3-4	reg – imm	×	×	×	×	×	×
		mem, imm	1	0	0	0	0	0	S	W	mod		1	1	1	mem			3-6	(mem) – imm	×	×	×	×	×	×
		acc, imm	0	0	1	1	1	1	0	W									2-3	If W = 0 : AL – imm If W = 1 : AW – imm	×	×	×	×	×	×
Complement operation instructions	NOT	reg	1	1	1	1	0	1	1	W	1	1	0	1	0	reg			2	reg ← $\overline{\text{reg}}$						
		mem	1	1	1	1	0	1	1	W	mod		0	1	0	mem			2-4	(mem) ← ($\overline{\text{mem}}$)						
	NEG	reg	1	1	1	1	0	1	1	W	1	1	0	1	1	reg			2	reg ← $\overline{\text{reg}}$ + 1	×	×	×	×	×	×
		mem	1	1	1	1	0	1	1	W	mod		0	1	1	mem			2-4	(mem) ← ($\overline{\text{mem}}$) + 1	×	×	×	×	×	×
Logical operation instructions	TEST	reg, reg'	1	0	0	0	0	1	1	W	1	1	reg				reg'		2	reg ∧ reg'	U	0	0	×	×	×
		mem, reg reg, mem	1	0	0	0	0	1	0	W	mod		reg			mem			2-4	(mem) ∧ reg	U	0	0	×	×	×
		reg, imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg			3-4	reg ∧ imm	U	0	0	×	×	×
		mem, imm	1	1	1	1	0	1	1	W	mod		0	0	0	mem			3-6	(mem) ∧ imm	U	0	0	×	×	×
		acc, imm	1	0	1	0	1	0	0	W									2-3	If W = 0 : AL ∧ imm8 If W = 1 : AW ∧ imm16	U	0	0	×	×	×

Table A-1. List of Mnemonics (15/33)

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags					
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Logical operation instructions	AND	reg, reg'	0 0 1 0 0 0 1 W / 1 1 reg reg'	2	reg ← reg ∧ reg'	U	0	0	×	×	×
		mem, reg	0 0 1 0 0 0 0 W / mod reg mem	2-4	(mem) ← (mem) ∧ reg	U	0	0	×	×	×
		reg, mem	0 0 1 0 0 0 1 W / mod reg mem	2-4	reg ← reg ∧ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W / 1 1 1 0 0 reg	3-4	reg ← reg ∧ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W / mod 1 0 0 mem	3-6	(mem) ← (mem) ∧ imm	U	0	0	×	×	×
		acc, imm	0 0 1 0 0 1 0 W	2-3	If W = 0 : AL ← AL ∧ imm8 If W = 1 : AW ← AW ∧ imm16	U	0	0	×	×	×
	OR	reg, reg'	0 0 0 0 1 0 1 W / 1 1 reg reg'	2	reg ← reg ∨ reg'	U	0	0	×	×	×
		mem, reg	0 0 0 0 1 0 0 W / mod reg mem	2-4	(mem) ← (mem) ∨ reg	U	0	0	×	×	×
		reg, mem	0 0 0 0 1 0 1 W / mod reg mem	2-4	reg ← reg ∨ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W / 1 1 0 0 1 reg	3-4	reg ← reg ∨ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W / mod 0 0 1 mem	3-6	(mem) ← (mem) ∨ imm	U	0	0	×	×	×
		acc, imm	0 0 0 0 1 1 0 W	2-3	If W = 0 : AL ← AL ∨ imm8 If W = 1 : AW ∨ imm16	U	0	0	×	×	×
	XOR	reg, reg'	0 0 1 1 0 0 1 W / 1 1 reg reg'	2	reg ← reg ⊻ reg'	U	0	0	×	×	×
		mem, reg	0 0 1 1 0 0 0 W / mod reg mem	2-4	(mem) ← (mem) ⊻ reg	U	0	0	×	×	×
		reg, mem	0 0 1 1 0 0 1 W / mod reg mem	2-4	reg ← reg ⊻ (mem)	U	0	0	×	×	×
		reg, imm	1 0 0 0 0 0 0 W / 1 1 1 1 0 reg	3-4	reg ← reg ⊻ imm	U	0	0	×	×	×
		mem, imm	1 0 0 0 0 0 0 W / mod 1 1 0 mem	3-6	(mem) ← (mem) ⊻ imm	U	0	0	×	×	×
		acc, imm	0 0 1 1 0 1 0 W	2-3	If W = 0 : AL ← AL ⊻ imm8 If W = 1 : AW ← AW ⊻ imm16	U	0	0	×	×	×

Table A-1. List of Mnemonics (16/33)

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC CY V P S Z
Bit manipulation instructions	TEST1	reg8, CL	0 0 0 1 0 0 0 0 / 1 1 0 0 0 reg	3	reg8 bit NO.CL = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		mem8, CL	0 0 0 0 / mod 0 0 0 mem	3-5	(mem8) bit NO.CL = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		reg16, CL	0 0 0 1 / 1 1 0 0 0 reg	3	reg16 bit NO.CL = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		mem16, CL	0 0 0 1 / mod 0 0 0 mem	3-5	(mem16) bit NO.CL = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		reg8, imm3	1 0 0 0 / 1 1 0 0 0 reg	4	reg8 bit NO.imm3 = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		mem8, imm3	1 0 0 0 / mod 0 0 0 mem	4-6	(mem8) bit NO.imm3 = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		reg16, imm4	1 0 0 1 / 1 1 0 0 0 reg	4	reg16 bit NO. imm4 = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
		mem16, imm4	1 0 0 1 / mod 0 0 0 mem	4-6	(mem16) bit NO. imm4 = 0 : Z ← 1 = 1 : Z ← 0	U 0 0 U U ×
	NOT1	reg8, CL	0 1 1 0 / 1 1 0 0 0 reg	3	reg8 bit NO. CL ← $\overline{\text{reg8 bit NO.CL}}$	
		mem8, CL	0 1 1 0 / mod 0 0 0 mem	3-5	(mem8) bit NO.CL ← $\overline{\text{(mem8) bit NO.CL}}$	
		reg16, CL	0 1 1 1 / 1 1 0 0 0 reg	3	reg16 bit NO.CL ← $\overline{\text{reg16 bit NO.CL}}$	
		mem16, CL	0 1 1 1 / mod 0 0 0 mem	3-5	(mem16) bit NO.CL ← $\overline{\text{(mem16) bit NO.CL}}$	
		reg8, imm3	1 1 1 0 / 1 1 0 0 0 reg	4	reg8 bit NO.imm3 ← $\overline{\text{reg8 bit NO.imm3}}$	
		mem8, imm3	1 1 1 0 / mod 0 0 0 mem	4-6	(mem8) bit NO.imm3 ← $\overline{\text{(mem8) bit NO.imm3}}$	
		reg16, imm4	1 1 1 1 / 1 1 0 0 0 reg	4	reg16 bit NO.imm4 ← $\overline{\text{reg16 bit NO.imm4}}$	
		mem16, imm4	1 1 1 1 / mod 0 0 0 mem	4-6	(mem16) bit NO.imm4 ← $\overline{\text{(mem16) bit NO.imm4}}$	
			2nd byte[Note] / 3rd byte[Note]		**Note** 1st byte = 0FH	
	NOT1	CY	1 1 1 1 0 1 0 1	1	CY ← $\overline{\text{CY}}$	×

Table A-1. List of Mnemonics (17/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Bit manipulation instructions	CLR1	reg8, CL	0	0	0	1	0	0	1	0	1	1	0	0	0	reg			3	reg8 bit NO.CL ← 0						
		mem8, CL					0	0	1	0	mod		0	0	0	mem			3-5	(mem8) bit NO.CL ← 0						
		reg16, CL					0	0	1	1	1	1	0	0	0	reg			3	reg16 bit NO.CL ← 0						
		mem16, CL					0	0	1	1	mod		0	0	0	mem			3-5	(mem16) bit NO.CL ← 0						
		reg8, imm3					1	0	1	0	1	1	0	0	0	reg			4	reg8 bit NO.imm3 ← 0						
		mem8, imm3					1	0	1	0	mod		0	0	0	mem			4-6	(mem8) bit NO.imm3 ← 0						
		reg16, imm4					1	0	1	1	1	1	0	0	0	reg			4	reg16 bit NO. imm4 ← 0						
		mem16, imm4					1	0	1	1	mod		0	0	0	mem			4-6	(mem16) bit NO. imm4 ← 0						
			2nd byte[Note]								3rd byte[Note]									**Note** 1st byte = 0FH						
		CY	1	1	1	1	1	0	0	0									1	CY ← 0		0				
		DIR	1	1	1	1	1	1	0	0									1	DIR ← 0						
	SET1	reg8, CL	0	0	0	1	0	1	0	0	1	1	0	0	0	reg			3	reg8 bit NO.CL ← 1						
		mem8, CL					0	1	0	0	mod		0	0	0	mem			3-5	(mem8) bit NO.CL ← 1						
		reg16, CL					0	1	0	1	1	1	0	0	0	reg			3	reg16 bit NO.CL ← 1						
		mem16, CL					0	1	0	1	mod		0	0	0	mem			3-5	(mem16) bit NO.CL ← 1						
		reg8, imm3					1	1	0	0	1	1	0	0	0	reg			4	reg8 bit NO.imm3 ← 1						
		mem8, imm3					1	1	0	0	mod		0	0	0	mem			4-6	(mem8) bit NO.imm3 ← 1						
		reg16, imm4					1	1	0	1	1	1	0	0	0	reg			4	reg16 bit NO. imm4 ← 1						
		mem16, imm4					1	1	0	1	mod		0	0	0	mem			4-6	(mem16) bit NO.imm4 ← 1						
			2nd byte[Note]								3rd byte[Note]									**Note** 1st byte = 0FH						
		CY	1	1	1	1	1	0	0	1									1	CY ← 1		1				
		DIR	1	1	1	1	1	1	0	1									1	DIR ← 1						

Table A-1. List of Mnemonics (18/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Bit manipulation instructions	BSCH	reg8	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	0	3	<1> CL ← 0 <2> [When reg8 bit No. CL = 0] When CL<7: CL ← CL+1, operation is executed from <2> When CL = 7: Z ← 1 [When reg8 bit No. CL = 1] Z ← 0	U	U	U	U	U	×
			1	1	0	0	0	reg																		
		mem8	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	0	3-5	<1> CL ← 0 <2> [When mem8 bit No. CL = 0] When CL<7: CL ← CL+1, operation is executed from <2> When CL = 7: Z ← 1 [When mem8 bit No. CL = 1] Z ← 0	U	U	U	U	U	×
			mod		0	0	0	mem																		
		reg16	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	1	3	<1> CL ← 0 <2> [When reg16 bit No. CL = 0] When CL<15: CL ← CL+1, operation is executed from <2> When CL = 15: Z ← 1 [When reg16 bit No. CL = 1] Z ← 0	U	U	U	U	U	×
			1	1	0	0	0	reg																		
		mem16	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0	1	3-5	<1> CL ← 0 <2> [When mem16 bit No. CL = 0] When CL<15: CL ← CL+1, operation is executed from <2> When CL = 15: Z ← 1 [When mem16 bit No. CL = 1] Z ← 0	U	U	U	U	U	×
			mod		0	0	0	mem																		

Table A-1. List of Mnemonics (19/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Shift instructions	SHL	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	0	reg			2	CY ← reg MSB, reg ← reg × 2 If reg MSB ≠ CY: V ← 1 If reg MSB = CY: V ← 0	U	×	×	×	×	×
		mem, 1	1	1	0	1	0	0	0	W	mod	1	0	0	mem				2-4	CY ← (mem) MSB, (mem) ← (mem) × 2 If (mem) MSB ≠ CY: V ← 1 If (mem) MSB = CY: V ← 0	U	×	×	×	×	×
		reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	0	reg			2	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← reg MSB, reg ← reg × 2 temp ← temp–1	U	×	U	×	×	×
		mem, CL	1	1	0	1	0	0	1	W	mod	1	0	0	mem				2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← (mem) MSB, (mem) ← (mem) × 2 temp ← temp–1	U	×	U	×	×	×
		reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	0	reg			3	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← reg MSB, reg ← reg × 2 temp ← temp–1	U	×	U	×	×	×
		mem, imm8	1	1	0	0	0	0	0	W	mod	1	0	0	mem				3-5	temp ← imm8, while temp ≠ 0 the following operation are repeated: CY ← (mem) MSB, (mem) ← (mem) × 2 temp ← temp–1	U	×	U	×	×	×

Table A-1. List of Mnemonics (20/33)

Instruction Group	Mnemonic	Operands	Operation Code																Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Shift instructions	SHR	reg, 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg			2	CY ← reg LSB, reg ← reg / 2 If reg MSB π bit after reg MSB: V ← 1 If reg MSB = bit after reg MSB: V ← 0	U	×	×	×	×	×
		mem, 1	1	1	0	1	0	0	0	W	mod		1	0	1	mem			2-4	CY ← (mem) LSB, (mem) ← (mem) / 2 If (mem) MSB ≠ bit after (mem) MSB: V ← 1 If (mem) MSB = bit after (mem) MSB: V ← 0	U	×	×	×	×	×
		reg, CL	1	1	0	1	0	0	1	W	1	1	1	0	1	reg			2	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 temp ← temp–1	U	×	U	×	×	×
		mem, CL	1	1	0	1	0	0	1	W	mod		1	0	1	mem			2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 temp ← temp–1	U	×	U	×	×	×
		reg, imm8	1	1	0	0	0	0	0	W	1	1	1	0	1	reg			3	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 temp ← temp–1	U	×	U	×	×	×
		mem, imm8	1	1	0	0	0	0	0	W	mod		1	0	1	mem			3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 temp ← temp–1	U	×	U	×	×	×

Table A-1. List of Mnemonics (21/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Shift instructions	SHRA	reg, 1	1	1	0	1	0	0	0	W	1	1	1	1	1		reg		2	CY ← reg LSB, reg ← reg / 2, V ← 0 MSB of operand is unchanged.	U	×	0	×	×	×
		mem, 1	1	1	0	1	0	0	0	W	mod		1	1	1		reg		2-4	CY ← (mem) LSB, (mem) ← (mem) / 2, V ← 0 MSB of operand is unchanged.	U	×	0	×	×	×
		reg, CL	1	1	0	1	0	0	1	W	1	1	1	1	1		reg		2	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 temp ← temp–1 MSB of operand is unchanged.	U	×	U	×	×	×
		mem, CL	1	1	0	1	0	0	1	W	mod		1	1	1		mem		2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 temp ← temp–1 MSB of operand is unchanged.	U	×	U	×	×	×
		reg, imm8	1	1	0	0	0	0	0	W	1	1	1	1	1		reg		3	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 temp ← temp–1 MSB of operand is unchanged.	U	×	U	×	×	×
		mem, imm8	1	1	0	0	0	0	0	W	mod		1	1	1		mem		3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 temp ← temp–1 MSB of operand is unchanged.	U	×	U	×	×	×

Table A-1. List of Mnemonics (22/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Rotate instructions	ROL	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	0	reg			2	CY ← reg MSB, reg ← reg × 2 + CY reg MSB ≠ CY: V ← 1 reg MSB = CY: V ← 0			×	×			
		mem, 1	1	1	0	1	0	0	0	W	mod	0	0	0	mem				2-4	CY ← (mem) MSB, (mem) ← (mem) × 2 + CY (mem) MSB ≠ CY: V ← 1 (mem) MSB = CY: V ← 0			×	×			
		reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	0	reg			2	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← reg MSB, reg ← reg × 2 + CY temp ← temp–1			×	U			
		mem, CL	1	1	0	1	0	0	1	W	mod	0	0	0	mem				2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← (mem) MSB, (mem) ← (mem) × 2 + CY temp ← temp–1			×	U			
		reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	0	reg			3	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← reg MSB, reg ← reg × 2 + CY temp ← temp–1			×	U			
		mem, imm8	1	1	0	0	0	0	0	W	mod	0	0	0	mem				3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← (mem) MSB, (mem) ← (mem) × 2 + CY temp ← temp–1			×	U			

Table A-1. List of Mnemonics (23/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Rotate instructions	ROR	reg, 1	1	1	0	1	0	0	0	W	1	1	0	0	1	reg			2	CY ← reg LSB, reg ← reg / 2 reg MSB ← CY reg MSB ≠ bit after reg MSB: V ← 1 reg MSB = bit after reg MSB: V ← 0		×	×			
		mem, 1	1	1	0	1	0	0	0	W	mod	0	0	1	mem				2-4	CY ← (mem) LSB, (mem) ← (mem) / 2 (mem) MSB ← CY (mem) MSB ≠ bit after (mem) MSB: V ← 1 (mem) MSB = bit after (mem) MSB: V ← 0		×	×			
		reg, CL	1	1	0	1	0	0	1	W	1	1	0	0	1	reg			2	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 reg MSB ← CY temp ← temp–1		×	U			
		mem, CL	1	1	0	1	0	0	1	W	mod	0	0	1	mem				2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 (mem) MSB ← CY temp ← temp–1		×	U			
		reg, imm8	1	1	0	0	0	0	0	W	1	1	0	0	1	reg			3	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← reg LSB, reg ← reg / 2 reg MSB ← CY temp ← temp–1		×	U			
		mem, imm8	1	1	0	0	0	0	0	W	mod	0	0	1	mem				3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: CY ← (mem) LSB, (mem) ← (mem) / 2 (mem) MSB ← CY temp ← temp–1		×	U			

Table A-1. List of Mnemonics (24/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Rotate instructions	ROLC	reg, 1	1	1	0	1	0	0	0	W	1	1	0	1	0	reg			2	tmpcy ← CY, CY ← reg MSB reg ← reg × 2 + tmpcy reg MSB ≠ CY: V ← 1 reg MSB = CY: V ← 0			×	×				
		mem, 1	1	1	0	1	0	0	0	W	mod		0	1	0	mem			2-4	tmpcy ← CY, CY ← (mem) MSB (mem) ← (mem) × 2 + tmpcy (mem) MSB ≠ CY: V ← 1 (mem) MSB = CY: V ← 0			×	×				
		reg, CL	1	1	0	1	0	0	1	W	1	1	0	1	0	reg			2	temp ← CY, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← reg MSB reg ← reg × 2 + tmpcy temp ← temp–1			×	U				
		mem, CL	1	1	0	1	0	0	1	W	mod		0	1	0	mem			2-4	temp ← CY, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← (mem) MSB (mem) ← (mem) × 2 + tmpcy temp ← temp–1			×	U				
		reg, imm8	1	1	0	0	0	0	0	W	1	1	0	1	0	reg			3	temp ← imm8, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← reg MSB reg ← reg × 2 + tmpcy temp ← temp–1			×	U				
		mem, imm8	1	1	0	0	0	0	0	W	mod		0	1	0	mem			3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← (mem) MSB (mem) ← (mem) × 2 + tmpcy temp ← temp–1			×	U				

Table A-1. List of Mnemonics (25/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Rotate instructions	RORC	reg, 1	1	1	0	1	0	0	0	W	1 1 0 1 1 reg	2	tmpcy ← CY, CY ← reg LSB reg ← reg / 2 reg MSB ← tmpcy reg MSB ≠ bit after reg MSB: V ← 1 reg MSB = bit after reg MSB: V ← 0		×	×			
		mem, 1	1	1	0	1	0	0	0	W	mod 0 1 1 mem	2-4	tmpcy ← CY, CY ← (mem) LSB (mem) ← (mem) / 2 (mem) MSB ← tmpcy (mem) MSB ≠ bit after (mem) MSB: V ← 1 (mem) MSB = bit after (mem) MSB: V ← 0		×	×			
		reg, CL	1	1	0	1	0	0	1	W	1 1 0 1 1 reg	2	temp ← CL, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← reg LSB reg ← reg / 2 reg MSB ← tmpcy temp ← temp–1		×	U			
		mem, CL	1	1	0	1	0	0	1	W	mod 0 1 1 mem	2-4	temp ← CL, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← (mem) LSB (mem) ← (mem) / 2 (mem) MSB ← tmpcy temp ← temp–1		×	U			
		reg, imm8	1	1	0	0	0	0	0	W	1 1 0 1 1 reg	3	temp ← imm8, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← reg LSB reg ← reg / 2 reg MSB ← tmpcy temp ← temp–1		×	U			
		mem, imm8	1	1	0	0	0	0	0	W	mod 0 1 1 mem	3-5	temp ← imm8, while temp ≠ 0 the following operations are repeated: tmpcy ← CY, CY ← (mem) LSB (mem) ← (mem) / 2 (mem) MSB ← tmpcy temp ← temp–1		×	U			

Table A-1. List of Mnemonics (26/33)

Instruction Group	Mnemonic	Operands	Operation Code														Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Subroutine control instructions	CALL	near-proc	1	1	1	0	1	0	0	0									3	(SP–1, SP–2) ← PC, SP ← SP–2 PC ← PC + disp						
		regptr16	1	1	1	1	1	1	1	1	1	1	0	1	0	reg			2	(SP–1, SP–2) ← PC, PC ← regptr16 SP ← SP–2						
		memptr16	1	1	1	1	1	1	1	1	mod	0	1	0	mem				2-4	(SP–1, SP–2) ← PC, SP ← SP–2 PC ← (memptr16)						
		far-proc	1	0	0	1	1	0	1	0									5	(SP–1, SP–2) ← PS, (SP–3, SP–4) ← PC SP ← SP–4 PS ← seg, PC ← offset						
		memptr32	1	1	1	1	1	1	1	1	mod	0	1	1	mem				2-4	(SP–1, SP–2) ← PS, (SP–3, SP–4) ← PC SP ← SP–4 PS ← (memptr32 + 2), PC ← (memptr32)						
	RET		1	1	0	0	0	0	1	1									1	PC ← (SP + 1, SP) SP ← SP + 2						
		pop-value	1	1	0	0	0	0	1	0									3	PC ← (SP + 1, SP) SP ← SP + 2, SP ← SP + pop-value						
			1	1	0	0	1	0	1	1									1	PC ← (SP + 1, SP) PS ← (SP + 3, SP + 2) SP ← SP + 4						
		pop-value	1	1	0	0	1	0	1	0									3	PC ← (SP + 1, SP) PS ← (SP + 3, SP + 2) SP ← SP + 4, SP ← SP + pop-value						

Table A-1. List of Mnemonics (27/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Stack manipulation instructions	PUSH	mem16	1	1	1	1	1	1	1	1	2-4	(SP–1, SP–2) ← (mem16) SP ← SP–2						
		reg16	0	1	0	1	0	reg			1	(SP–1, SP–2) ← reg16 SP ← SP–2						
		sreg	0	0	0	sreg		1	1	0	1	(SP–1, SP–2) ← sreg SP ← SP–2						
		PSW	1	0	0	1	1	1	0	0	1	(SP–1, SP–2) ← PSW SP ← SP–2						
		R	0	1	1	0	0	0	0	0	1	Push registers on the stack						
		imm	0	1	1	0	1	0	S	0	2-3	(SP–1, SP–2) ← imm SP ← SP–2, sign extension if S = 1						
	POP	mem16	1	0	0	0	1	1	1	1	2-4	SP ← SP + 2 (mem16) ← (SP–1, SP–2)						
		reg16	0	1	0	1	1	reg			1	SP ← SP + 2 reg16 ← (SP–1, SP–2)						
		sreg	0	0	0	sreg		1	1	1	1	SP ← SP + 2 sreg ← (SP–1, SP–2) sreg: SS, DS0, DS1						
		PSW	1	0	0	1	1	1	0	1	1	SP ← SP + 2 PSW ← (SP–1, SP–2)	R	R	R	R	R	R
		R	0	1	1	0	0	0	0	1	1	Pop registers from the stack						
	PREPARE	imm16, imm8	1	1	0	0	1	0	0	0	4	Prepare New Stack Frame						
	DISPOSE		1	1	0	0	1	0	0	1	1	Dispose of Stack Frame						

Table A-1. List of Mnemonics (28/33)

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC CY V P S Z
Branch instructions	BR	near-label	1 1 1 0 1 0 0 1	3	PC ← PC + disp	
		short-label	1 1 1 0 1 0 1 1	2	PC ← PC + ext – disp8	
		regptr16	1 1 1 1 1 1 1 1 / 1 1 1 0 0 reg	2	PC ← regptr16	
		memptr16	1 1 1 1 1 1 1 1 / mod 1 0 0 mem	2-4	PC ← (memptr16)	
		far-label	1 1 1 0 1 0 1 0	5	PS ← Seg PC ← offset	
		memptr32	1 1 1 1 1 1 1 1 / mod 1 0 1 mem	2-4	PS ← (memptr32 + 2) PC ← (memptr32)	
Conditional branch instructions	BV	short-label	0 1 1 1 0 0 0 0	2	if V = 1 PC ← PC + ext–disp8	
	BNV	short-label	0 0 0 1	2	if V = 0 PC ← PC + ext–disp8	
	BC BL	short-label	0 0 1 0	2	if CY = 1 PC ← PC + ext–disp8	
	BNC BNL	short-label	0 0 1 1	2	if CY = 0 PC ← PC + ext–disp8	
	BE BZ	short-label	0 1 0 0	2	if Z = 1 PC ← PC + ext–disp8	
	BNE BNZ	short-label	0 1 0 1	2	if Z = 0 PC ← PC + ext–disp8	
	BNH	short-label	0 1 1 0	2	if CY∧ Z = 1 PC ← PC + ext–disp8	
	BH	short-label	0 1 1 1	2	if CY∧ Z = 0 PC ← PC + ext–disp8	
	BN	short-label	1 0 0 0	2	if S = 1 PC ← PC + ext–disp8	
	BP	short-label	1 0 0 1	2	if S = 0 PC ← PC + ext–disp8	
	BPE	short-label	1 0 1 0	2	if P = 1 PC ← PC + ext–disp8	
	BPO	short-label	1 0 1 1	2	if P = 0 PC ← PC + ext–disp8	
	BLT	short-label	1 1 0 0	2	if S∀ V = 1 PC ← PC + ext–disp8	

Table A-1. List of Mnemonics (29/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Conditional branch instructions	BGE	short-label	0	1	1	1	1	1	0	1	2	if S∀V = 0 PC ← PC + ext–disp8						
	BLE	short-label					1	1	1	0	2	if (S∀V)∨Z = 1 PC ← PC + ext–disp8						
	BGT	short-label					1	1	1	1	2	if (S∀V)∨Z = 0 PC ← PC + ext–disp8						
	DBNZNE	short-label	1	1	1	0	0	0	0	0	2	CW = CW–1 if Z = 0 and CW ≠ 0 PC ← PC + ext–disp8						
	DBNZE	short-label					0	0	0	1	2	CW = CW–1 if Z = 1 and CW ≠ 0 PC ← PC + ext–disp8						
	DBNZ	short-label					0	0	1	0	2	CW = CW–1 if CW ≠ 0 PC ← PC + ext–disp8						
	BCWZ	short-label					0	0	1	1	2	if CW = 0 PC ← PC + ext–disp8						
	BTCLR[Note]	sfr, imm3, short-label	0	0	0	0	1	1	1	1 sfr disp8	1	0 0 1 1 1 1 0 0 imm3	5	If (sfr) bit No.imm3 = 1 PC ← PC + ext–disp8, (sfr) bit No.imm3 ← 0						

Note This instruction is not provided in the V20 or V30.

Table A-1. List of Mnemonics (30/33)

Instruction Group	Mnemonic	Operands	Operation Code								Bytes	Operation	Flags					
			7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Interrupt instructions	BRK	3	1	1	0	0	1	1	0	0	1	(SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)						
		imm8 (π 3)	1	1	0	0	1	1	0	1	2	(SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0 PS ← (n × 4 + 3, n × 4 + 2), PC ← (n × 4 + 1, n × 4) n = imm8						
	BRKV		1	1	0	0	1	1	1	0	1	If V = 1 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)						
	RETI		1	1	0	0	1	1	1	1	1	PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	R	R	R	R	R	R
	RETRBI[Note]		0	0	0	0	1	1	1	1	2	PC ← Value of the PC save area on the register bank currently selected PSW ← Value of the PSW save area on the register bank currently selected	R	R	R	R	R	R
	FINT[Note]		0	0	0	0	1	1	1	1	2	Indicates to interrupt controller incorporated in CPU that interrupt servicing has ended.						
	CHKIND	reg16, mem32	0	1	1	0	0	0	1	0	2-4	If (mem32) > reg16 or (mem32 + 2) < reg16 (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC, SP ← SP–6 IE ← 0, BRK ← 0 PS ← (23, 22), PC ← (21, 20)						

Note This instruction is not provided in the V20 or V30.

Table A-1. List of Mnemonics (31/33)

Instruction Group	Mnemonic	Operands	Operation Code		Bytes	Operation	Flags					
			7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Bank switching	BRKCS[Note 1]	reg16	0 0 0 0 1 1 1 1	0 0 1 0 1 1 0 1	3	temp ← PSW RB3-0 ← reg16 low-order 4 bits, IE ← 0, BRK ← 0, PSW save area on the register bank newly selected ← temp PC save area on the register bank newly selected ← PC PC ← Vector PC on the register bank newly selected						
			1 1 0 0 0 reg									
	TSKSW[Note 1]	reg16	0 0 0 0 1 1 1 1	1 0 0 1 0 1 0 0	3	PSW save area on the register bank currently selected ← PSW PC save area on the register bank currently selected ← PC RB3-0 ← reg16 low-order 4 bits, PSW ← Value of the PSW save area on the register bank newly selected PC ← Value of the PC save area on the register bank newly selected	×	×	×	×	×	×
			1 1 1 1 1 reg									
CPU control instructions	HALT		1 1 1 1 0 1 0 0		1	CPU Halt						
	STOP[Note 1]		0 0 0 0 1 1 1 1	1 0 0 1 1 1 1 0	2	CPU Stop						
	POLL		1 0 0 1 1 0 1 1		1	Poll and wait						
	DI		1 1 1 1 1 0 1 0		1	IE ← 0						
	EI		1 1 1 1 1 0 1 1		1	IE ← 1						
	BUSLOCK		1 1 1 1 0 0 0 0		1	Bus Lock Prefix						
	FPO1	fp-op	1 1 0 1 1 X X X	1 1 Y Y Y Z Z Z	2	(SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC–x[Note 3], SP ← SP–6 IE ← 0, BRK ← 0 PC ← (01DH, 01CH), PS ← (01FH, 01EH)						
		fp-op, mem	1 1 0 1 1 X X X	mod Y Y Y mem	2-4							
	FPO2	fp-op	0 1 1 0 0 1 1 X	1 1 Y Y Y Z Z Z	2							
		fp-op, mem	0 1 1 0 0 1 1 X	mod Y Y Y mem	2-4							
	NOP		1 0 0 1 0 0 0 0		1	No Operation						
Segment override prefixes	**Note 2**		0 0 1 sreg 1 1 0		1	Segment override prefixes						

Notes 1. This instruction is not provided in the V20 or V30.

2. Four kinds: DS0:, DS1:, PS:, SS:

3. x: Instruction byte number + Prefix number

Table A-1. List of Mnemonics (32/33)

Instructions not provided in the V25 and V35

Instruction Group	Mnemonic	Operands	Operation Code	Bytes	Operation	Flags
			7 6 5 4 3 2 1 0 / 7 6 5 4 3 2 1 0			AC CY V P S Z
Additional instructions	PUSH	DS2	0 0 0 0 1 1 1 1 / 0 1 1 1 1 1 1 0	2	(SP–1, SP–2) ← DS2, SP ← SP–2	
		DS3/VPC	0 0 0 0 1 1 1 1 / 0 1 1 1 0 1 1 0	2	(SP–1, SP–2) ← DS3/VPC, SP ← SP–2	
	POP	DS2	0 0 0 0 1 1 1 1 / 0 1 1 1 1 1 1 1	2	SP ← SP + 2, DS2 ← (SP–1, SP–2)	
		DS3/VPC	0 0 0 0 1 1 1 1 / 0 1 1 1 0 1 1 1	2	SP ← SP + 2, DS3/VPC ← (SP–1, SP–2)	
	DS2:		0 1 1 0 0 0 1 1	1	Extended segment override prefix	
	DS3:		1 1 0 1 0 1 1 0	1	Extended segment override prefix	
	MOV	xsreg, reg16	1 0 0 0 1 1 1 0 / 1 1 1 xsreg reg	2	xreg ← reg16 xsreg: DS2, DS3/VPC	
		xsreg, mem16	1 0 0 0 1 1 1 0 / mod 1 xsreg mem	2-4	xreg ← (mem16) xsreg: DS2, DS3/VPC	
		reg16, xsreg	1 0 0 0 1 1 0 0 / 1 1 1 xsreg reg	2	reg16 ← xsreg xsreg: DS2, DS3/VPC	
		mem16, xsreg	1 0 0 0 1 1 0 0 / mod 1 xsreg mem	2-4	(mem16) ← xsreg xsreg: DS2, DS3/VPC	
		DS2, reg16, mem32	0 0 0 0 1 1 1 1 / 0 0 1 1 1 1 1 0 mod reg mem	3-5	reg16 ← (mem32) DS2 ← (mem32 + 2)	
		DS3, reg16, mem32	0 0 0 0 1 1 1 1 / 0 0 1 1 0 1 1 0 mod reg mem	3-5	reg16 ← (mem32) DS3 ← (mem32 + 2)	
	IRAM:		1 1 1 1 0 0 0 1	1	Override prefix for register file	
	RSTWDT	imm8, $\overline{imm8'}$	0 0 0 0 1 1 1 1 / 1 0 0 1 0 1 1 0 imm8 / $\overline{imm8'}$	4	If imm8 = $\overline{imm8'}$ WDM register ← imm8 If imm8 ≠ $\overline{imm8'}$ (SP–1, SP–2) ← PSW, (SP–3, SP–4) ← PS, (SP–5, SP–6) ← PC–x[Note], SP ← SP–6 IE ← 0, BRK ← 0 PC ← (20H, 21H), PS ← (22H, 23H)	
	BTCLRL	sfrl, imm3, short-label	0 0 0 0 1 1 1 1 / 1 0 0 1 1 1 0 1 sfrl / imm3 disp8	5	If (sfrl) bit No.imm3 = 1 PC ← PC + ext – dep8. (sfrl) bit No.imm3 ← 0	

Note x: Instruction byte number + Prefix number

Table A-1. List of Mnemonics (33/33)

Queue Manipulation Instructions

Instruction Group	Mnemonic	Operation Code																Bytes	Operation	Flags					
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Queue manipulation instructions	QHOUT	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	4	Removes block at head of queue.	U	U	U	U	U	×
	QOUT	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	1	4	Removes any queue block.	U	U	U	U	U	×
	QTIN	0	0	0	0	1	1	1	1	0	1	1	1	0	0	1	0	4	Queues block at queue.						

Dedicated Fax Instructions

Instruction Group	Mnemonic	Operation Code																Bytes	Operation	Flags					
		7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Dedicated fax instructions	ALBIT	0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	0	2	If CH + CL ≥ 16: BW, DW → DS1: IY output to transmit buffer If CH + CL < 16: CH + CL → CH, BW, DW → BW Part exceeding 16 bits: CH + CL – 16 → CH, BW, DW → BW	U	×	U	U	U	×
	COLTRP	0	0	0	0	1	1	1	1	1	0	0	1	1	0	1	1	2	Stores 1 line pixel data change point information in change point table (start white run length).						
	MHENC	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1	1	2	Generates MH codes from change point table.	U	×	U	U	U	×
	MRENC	0	0	0	0	1	1	1	1	1	0	0	1	0	1	1	1	2	Generates MR codes from change point table.	U	×	U	U	U	×
	SCHEOL	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0	2	Extracts “EOL” from MH/MR codes.	U	×	U	U	U	×
	GETBIT	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	1	2	Extracts start bit of pixel data and sets in CY flag.	U	×	U	U	U	×
	MHDEC	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	0	2	Generates change point table from MH codes.	U	×	U	U	U	×
	MRDEC	0	0	0	0	1	1	1	1	0	1	1	1	1	1	0	1	2	Generates change point table from MR codes.	U	×	U	U	U	×
	CNVTRP	0	0	0	0	1	1	1	1	0	1	1	1	1	0	1	0	2	Converts change point information for one line of change point table into pixel data.						

APPENDIX B OPERATING PRECAUTIONS

B.1 Interrupt Function

(1) SFR access using macro service

Set the low-order 16 bits of the SFR address to the SFR address specification pointer of the macro service channel. Set 1 to all the high-order 7 bits.

(2) Interrupt mode control register manipulation

Be sure to set the interrupt mode control register once from an initialization after resetting even if change is not required. This is necessary to prevent system malfunction caused by an incorrect writing operation. Be sure to set the register in interrupt disabled state (DI state) for manipulation.

(3) Interrupt request generation at the end of macro service

During a vector interrupt (or register bank switching) processing which is generated at the end of the macro service, if the same interrupt request is generated again, the IF flag of the interrupt request control register is set to 1. In this state, set the MS/$\overline{\text{INT}}$ bit of the interrupt request control register to 1. The macro service can be executed at once. After the macro service is completed, the interrupt processing is executed again.

(4) If an interrupt is not generated

If no interrupt is generated when it should have, first check the following points.

(a) Is the status EI?

Check whether the EI instruction has been executed. If the IE flag of the PSW register has been set (to "1"), the V55PI is currently in the EI status.

(b) Is the interrupt masked?

In this case, reset (to "0") bit 6 of the interrupt request control register (ICxx) to cancel the mask.

(c) Has an interrupt request been generated?

If bit 7 (IF) of the interrupt request control register (ICxx) is set (to "1"), an interrupt request is generated. When this bit is "0", no interrupt request is generated. Check whether the interrupt request has been cleared by software.

(d) Is a high-priority interrupt being processed?

A bit corresponding to a high interrupt priority may be set to the in-service priority register (ISPR). In this case, the V55PI checks whether an interrupt of the current priority level of the set bit is currently being processed, and will not accept an interrupt with a lower priority level.

The bits of the ISPR register can be set only with the FINT instruction.

Therefore, check whether the FINT instruction is being executed during interrupt processing of the set priority level.

For details on the ISPR register, see **In-service priority register (ISPR)**.

(5) Cautions regarding FINT instruction

The FINT instruction must be executed immediately before the last RETI instruction during interrupt processing.

The FINT instruction clears the bit corresponding to the highest priority level of the bit set in the ISPR register. If the FINT instruction is not executed during the interrupt processing routine, interrupts with a priority level lower than that interrupt are not accepted.

Also, if the FINT instruction is executed while an interrupt is being processed, the V55PI becomes unable to manage priorities, and starts accepting also interrupts with a priority level lower than the interrupt currently being processed. Therefore, if the FINT instruction is executed while an interrupt is being processed, manage priority levels from the application.

For details on the FINT instruction, see **V55PI UM Instructions**.

B.2 DMA Function

(1) Restrictions on transfer object specification with main channel DMA controller

In the following DMA operating modes in which the main channel is used, note that the restrictions described below apply when the transfer object is an SFR or memory.

Intelligent DMA mode-1
Intelligent DMA mode-2
Next address specification mode
Memory-memory transfer mode

(a) SFRs which can be specified as transfer object

Only the SFRs shown below can be specified as the object of a transfer.

If another SFR (interrupt request control register, interrupt mask flag register, in-service priority register, DMA-related control register, etc.) is specified as the object of DMA transfer, subsequent normal operation cannot be guaranteed.

Special Function Register Names	Special Function Register Symbols
Ports, port 7 buffer	P0 to P8, P7L, P7H
Real-time output port	RTP
Timer compare registers	CM00, CM01, CM10, CM11, CM20, CM21, CM22, CM23, CM30, CM31
PWM register	PWM
UART transmit/receive buffers	TxB0, TxB1, RxB0, RxB1
Parallel interface buffer	PAB
A/D conversion result registers	ADCR0 to ADCR3

(b) Using the DMA function and macro service together

When the DMA function and macro service are executed together, do not specify the same memory block or same SFR as the transfer object for both processing.

(c) Cautions on using the external 8-bit bus mode

When using the external 8-bit bus mode, DMA transfer between the 16-bit SFR (See **Table 3-5 List of Special Function Registers**) and the external memory cannot be executed (See **Table 6-7 External Data Bus Width and DMA Transfer Objects**).

(2) Reading the register during DMA transfer

In the next address specification mode and memory-to-memory transfer mode, the contents of the TCM register is loaded to the TC register immediately before the next DMA cycle is executed. Therefore, if the TC register is read immediately after a borrow occurs, the value is read out as it is.

(3) Changing the TC register value during DMA interrupt processing

To change a TC value during the DMA interrupt processing in the next address specification mode, first disable the DMA transfer before changing the value. Moreover, be sure to perform either of the following manipulations.

- Write a TC value to be newly set to the TCM register.
- Write "1" into the END/F flag of the DMAS register (As it is returned to the initial state, the TC value is counted down as it is).

If the next DMA request is generated before the DMA transfer is disabled, the TCM value is loaded to the TC and is executed.

(4) Cautions on DMA status registers (DMAS)

Even if the DMA transfer mode is switched, the contents of the DMAS register is not initialized. Initialize it as necessary after the mode switching.

The 1-bit manipulation instruction cannot be used for this register. Write data in which only bits to be reset is set to the register 1 by the 8-bit manipulation instruction. Note that the bit test instruction can be used.

(5) Transfer mode in refresh cycle automatic insertion enable

When an I/O-to-memory DMA transfer is executed in a condition where the refresh cycle automatic insertion is enable (RFM register RFEN bit = 1), use the single transfer mode. If the demand release mode is specified, the final DMA transfer cycle following the rising of DMARQ input may not be generated correctly.

(6) UART receive interrupt request used as a DMA transfer request source

When a UART reception is executed using the DMA function, even if an error occurs, the DMA transfer is executed from the receive buffer (RxB) to an external memory according to the receive interrupt request. In this case, take precautions against the receive error interrupt which is generated after the DMA transfer.

Data in which a receive error has occurred is stored in the address represented as MAR + TC + 1 (decrement) or MAR-TC-1 (increment) in the external memory space.

When a receive data in which an error has occurred is manipulated in the receive error interruption, set the DMA to the disable state to prevent the next DMA transfer from being generated.

★ **(7) Cautions when using single-step mode**

If data read (IN instruction or INM instruction) from channel 1 input/output pointer (IOP1) during channel 0 DMA transfer is executed with the following settings, several cautions apply depending on the mode of channel 0.

These cautions do not apply when channel 0 DMA transfer is not performed.

(a) Settings

- Set single-step mode as the channel 0 transportation mode, and, at the same time, 2-channel operation mode, Memory-memory transfer mode, or Intelligent DMA mode-2 as the operation mode.
- Set Intelligent DMA mode-1 as the channel 1 operation mode.

(b) Cautions on each operation mode of channel 0

Channel 0 Operation mode	Channel 0 DCM0 Value	Applicable Cautions
2-channel operation mode (main channel)	000000H	<1>, <2>
2-channel operation mode (main channel)	Other than 000000H	<2>, <3>
2-channel operation mode (subchannel)	Does not depend on value	<2>, <3>, <4>
Memory-memory transfer mode	1FFFFFH	<1>, <2>
Memory-memory transfer mode	Other than 1FFFFFH	<2>, <3>
Intelligent DMA mode-2	Does not depend on value	<2>, <3>, <5>Note

Note Do not use it with the combination of channel 0 set to Intelligent DMA mode-2 (single-step mode) and channel 1 set to Intelligent DMA mode-1.

<1> Each time channel 1 reads IOP1, an illegal channel 0 INTD0 interrupt occurs. If channel 0 is set to 2-channel operation mode, set a value other than 000000H to DCM0, and if channel 0 is set to Memory-memory transfer mode, set a value other than 1FFFFFH to DCM0.

<2> After an INTD1 interrupt occurs with DCM1 = UDC1 during DMA transfer in Intelligent DMA mode-1 on channel 1, if data is read from the IOP1 area by the IN instruction in buffer-full status, UDC1 is decremented (-1), but the INTD1 interrupt that is supported to occur does not. Therefore, after INTD1 buffer-full interrupt occurs with DCM1 = UDC1 during DMA transfer in Intelligent DMA mode-1 on channel 1, execute the INTD1 interrupt which normally occurs when read of IOP1 in buffer-full status is executed (the INTD1 interrupt that occurs when UDC1 is decremented (-1) does not fulfill any particular role).

<3> The buffer-full flag END/F (bit 2 of DMAS) is set (1) with DCM1 = UDC1 during DMA transfer in Intelligent DMA mode-1 on channel 1. Next, if data is read from the IOP1 area by the IN instruction in buffer-full status, UDC1 is decremented (-1), but the END/F flag that normally should be reset (0) is not reset. Therefore, at the same timing as <2>, the buffer-full flag END/F (bit 2 of DMAS) must be set (1) with DCM1 = UDC1 during DMA transfer in Intelligent DMA mode-1 on channel 1, and next, the END/F flag must be reset (0) by software when the contents of IOP1 are read in buffer-full status.

<4> When the contents of IOP1 of channel 1 are read, UDC1 of channel 1 is decreased (-1). As a result of this decrement (-1), the buffer-empty flag EMP (bit 3 of DMAS) which should be set (1) is not set when borrow occurs from UDC1. Therefore, the buffer-empty flag EMP must not be used. (Buffer-empty can be detected by checking the value of UDC1 to see if it is 1FFFFFH).

<5> When borrow occurs at the same timing as <4>, channel 0 is not affected, but the DCM0 value is reloaded to UDC0 of channel 0 and the UDC0 value is changed. There is no way to avoid this.

★ **(8) Cautions when using intelligent DMA mode-1**

If the buffer-full flag END/F (bit 2 of DMAS) is reset (0) by an instruction (execution of write "1") immediately after borrow of terminal counter occurs (TC = 1FFFFFFH), the borrow information dissapears, and the DCM value cannot be reloaded to TC at the next IOP read or DMA transfer. For this reason, if DMA transfer to the address indicated by MAR + TC is performed subsequently, transfer will be done to the wrong address MAR + 1FFFFFH. Therefore, the END/F flag must not be reset (0) by an instruction ("1" must not be written).

B.3 Serial Interface Function (UART)

(1) Transmission disabled operation during transmission

If transmission disabled operation is performed during transmission by setting $\overline{\text{CTS}}$ to 1 or TxE to 0, the data being transmitted is transmitted to the end and the transmission is stopped when the data of the transmit shift register has been transmitted.

If the transmit buffer (TxB) contains data, it is not transmitted and is stored until the next transmission enable.

(2) Setting transmit interrupt

When transmission is enabled before the transmit data is written in the transmit buffer, the transmit interrupt request is generated immediately. Always set the transmit interrupt before enabling transmission.

(3) Reception interrupt accompanying reception error interrupt

When reception error interrupt is generated, the reception interrupt request is also generated following it. If the receive interrupt after the receive error interrupt is not to be acknowledged, clear the RxE bit of the UARTM register. The receive data in the receive buffer (RxB) is transmitted even when errors are generated. When clearing the RxE bit with the software, always read the receive buffer value to prevent overrun errors.

(4) Value setting in reception BRG register

With the UART, a value of 7 or more should be set in the reception BRG register in order to minimize phase misalignment between the serial clock and receive data.

(5) Reading transmit/receive BRG register

The transmit/receive BRG register should not be read during transmission/reception. Otherwise the shift clock cycle will be temporarily disrupted.

(6) Transmit/receive interrupt

In the V55P1, two transmit/receive generations are memorized in the interrupt controller. However, when the interrupt flag is cleared to erase the interrupt request with the software, the two interrupt requests will also be erased.

(7) If transfer is interrupted while in progress

If data transfer is interrupted while in progress, first check the following points.

<1> Is data set in the transfer buffer during transfer interrupt?

A transfer interrupt is generated when TxB becomes empty as a result of transfer data being transferred from the transmission buffer (TxB) to the shift register. Therefore, if setting transfer data with a transfer interrupt, be sure to write data to TxB during the transfer interrupt.

<2> Has the interrupt flag (IF) been cleared by software?
When setting transfer enable, an interrupt request is immediately generated.
Therefore, if you set the interrupt request control register (ICxx) after setting transfer enable, the interrupt flag may be cleared by error.
See **Section B.1 (4) If an interrupt is not generated**.

B.4 Serial Interface Function (CSI)

(1) Setting order in SBI mode

In the SBI mode, the following settings must be made before enabling reception (CSIM register CRxE bit = 1):

- Serial clock selection
- SBI mode serial data input/output pin selection

(2) Canceling receive data in IOE mode

If receive operation being performed is stopped in the IOE mode, the CTxE bit and CRxE bit of the CSIM register are both set to 0, the shift counter is reset, and the shift register (SIO) is returned to the initial state.

B.5 Timer Function

- **Count Timing of OT1 input event**

The sampling of the event count input pin (TI) is performed every $\phi/8$. The count precision is as follows.

Dividing ratio of system clock	Sampling time	
	fxx = 25 MHz	fxx = 32 MHz
ϕ = fxx/2	0.64 μs	0.50 μs
ϕ = fxx/4	1.28 μs	1.00 μs
ϕ = fxx/8	2.56 μs	2.00 μs
ϕ = fxx/16	5.12 μs	4.00 μs

fxx: Crystal oscillation frequency

<1> In the case of an application circuit in which the serial clock is input to the external event input pin TI and the communication speed is calculated, if a communication speed of 64 Kbps is used, a precision of at least 15.6 μs is required.

<2> Taking account of the sampling precision, the maximum input clock frequency is around 156 kHz in ϕ = fxx/2 mode.
However, a miscalculation will not result if TI < $\phi/8$ in the circuit (this is not very suitable for application circuits which require fast, real-time response to external event input).

B.6 Real-Time Output Port Function

When switching from "delay" mode to "no delay" mode during the delay counter operation, the output state of P70 to P73 is not changed until the P7L buffer value is transmitted to the real-time output port the next time.

To output the RTP0 through RTP3 and P70 through P73, manipulate the TRG bit of the RTPC register (See **10.4.2 Real-time output port control register (RTPC)**).

B.7 A/D Converter Function

(1) Executing the STOP instruction

When setting the STOP mode while the A/D converter is operating, always stop the A/D converter conversion operations. This is to prevent the sampling gate from being left open when the A/D converter is stopped during sampling operations. By setting the CS bit of the A/D converter mode register to 0, all conversion operations will be stopped even if conversion is being performed.

(2) A/D conversion sequence initialization

The A/D conversion sequence is initialized by effective edge input to the external trigger pin and by resetting ($1 \rightarrow 1$) of the TRG bit of the A/D converter mode register. However, set the period of the effective edge input to the external longer than one conversion operation period because conversion is not performed when the A/D conversion is performed by repeated effective edge input to the external pin, etc., if the effective edge input period is shorter than one conversion sequence period.

(3) Loss of conversion result

If the A/D conversion result is read by vector interrupt in the select mode of the A/D converter, the conversion results may be lost if the A/D conversion end interrupt is kept for a long time (160 clocks when the FR bit is 0 and above 120 clocks when the FR bit is 1) by other interrupt processes. As it is difficult to detect this loss of conversion results with the software, take the following measures to prevent such losses.

- Make the time of other interrupt processes sufficiently short compared to the A/D conversion time.
- Using multiple-interrupt function, acknowledge the A/D conversion end interrupt even while other interrupt services are performed.
- Perform A/D conversion end interrupt using macro service.

B.8 Codec Function

(1) Data transfer direction

In the codec instruction, the start bit of serial data such as pixel data, code data, etc. is always treated as a LSB side.

(2) Re-execution after instruction execution suspension

When re-executing codec instructions (SCHEOL, MHENC, MRENC, MHDEC, MRDEC) after an instruction execution is suspended, keep parameters that require 0 clear in the suspended state.

(3) Specifying data buffer size

Set an even value except zero to the reception data buffer size as the input data of codec instructions (SCHEOL, GETBIT, MHDEC, MRDEC). Also set an even value (except zero) to the pixel data buffer size of the COLTRP instruction.

(4) Adding the extended segment override prefix

When the extended segment override prefix (DS2:, DS3:) is added to codec instructions (COLTRP, SCHEOL, GETBIT, MHDEC, MRDEC, MHENC, MRENC, CNVTRP), 00H and 02H are referred to instead of 08H and 0EH as the offset value of the register bank which is specified by the AL register as an input parameter.

APPENDIX C REGISTER INDEX

[A]

A/D Conversion Result Register (ADCR0-ADCR3) ... 392, 400
A/D Converter Mode Register (ADM) ... 399

[C]

Capture Register (CT00, CT01, CT10) ... 349
Clocked Serial I/O Shift Register (SIO0, SIO1) ... 315
Clocked Serial Interface Mode Register (CSIM0, CSIM1) ... 305
Compare Register (CM00, CM01, CM10, CM11, CM20, CM21, CM22, CM23, CM30, CM31) ... 349

[D]

Delay Counter ... 380
DMA Compare Register (DCM0, DCM1) ... 244
DMA Control Register (DMAC0, DMAC1) ... 237
DMA Memory Address Register (MAR0, MAR1) ... 243
DMA Mode Register (DMAM0, DMAM1) ... 233
DMA Read/Write Pointer (DPTC0, DTPC1) ... 243
DMA Status Register (DMAS) ... 240
DMA Up/Down Counter (UDC0, UDC1) ... 243

[E]

External Interrupt Mode Register (INTM0, INTM1) ... 179

[I]

In-service Priority Register (ISPR) ... 178
Interrupt Mask Flag Register (MK0L, MK0H, MK1L, MK1H) ... 174
Interrupt Mode Control Register (IMC) ... 172
Interrupt Request Control Register (IC9-14, 16-32, 36, 37) ... 175

[M]

Memory Block Control Register (MBC) ... 90

[P]

Parallel Interface Acknowledge Interval Register (PAI1, PAI2) ... 332
Parallel Interface Buffer Register (PAB) ... 333
Parallel Interface Control Register 0 (PAC0) ... 323
Parallel Interface Control Register 1 (PAC1) ... 328
Parallel Interface Status Register (PAS) ... 330
Port 0 (P0) ... 407
Port 0 Mode Register (PM0) ... 407
Port 1 (P1) ... 408
Port 2 (P2) ... 409
Port 2 Mode Control Register (PMC2) ... 409
Port 2 Mode Register (PM2) ... 410

Port 3 (P3) ... 410
Port 3 Mode Control Register (PMC3) ... 411
Port 3 Mode Register (PM3) ... 411
Port 4 (P4) ... 411
Port 4 Mode Control Register (PMC4) ... 412
Port 4 Mode Register (PM4) ... 412
Port 5 (P5) ... 413
Port 5 Mode Control Register (PMC5) ... 413
Port 5 Mode Register (PM5) ... 414
Port 6 (P6) ... 414
Port 7 (P7) ... 415
Port 7 Buffer (Higher) (P7H) ... 369
Port 7 Buffer (Lower) (P7L) ... 369
Port 7 Mode Control Register (PMC7) ... 377, 416
Port 7 Mode Register (PM7) ... 416
Port 8 (P8) ... 417
Port 8 Mode Control Register (PMC8) ... 417
Port 8 Mode Register (PM8) ... 417
Port Read Control Register (PRDC) ... 403
Prescaler Register (PRS0, PRS1) ... 272
Processor Control Register (PRC) ... 426, 429
Programmable Wait Control Register (PWC0, PWC1) ... 94
Protocol Selection Register (ASP) ... 267
PWM Control Register (PWMC) ... 386
PWM Register (PWM) ... 386

[R]

Real-time Output Port (RTP) ... 369
Real-time Output Port Control Register (RTPC) ... 369, 378
Real-time Output Port Delay Specification Register (RTPD) ... 369, 380
Receive Baud Rate Generator Register (RxBRG0, RxBRG1) ... 273
Receive Buffer (RxB0, RxB1) ... 286, 315
Refresh Mode Register (RFM) ... 101

[S]

SBI Control Register (SBIC0) ... 308
Software Timer Counter (STC) ... 430
Software Timer Counter Compare Register (STMC) ... 430
Standby Control Register (STBC) ... 420

[T]

Terminal Counter (TC0, TC1) ... 242
Terminal Counter Modulo Register (TCM0, TCM1) ... 243
Timer Control Register (TMC0, TMC1) ... 369
Timer Output Control Register (TOC0, TOC1) ... 365
Timer Register (TM0-TM3) ... 349
Transmit Baud Rate Generator Register (TxBRG0, TxBRG1) ... 273

[U]
UART Mode Register (UARTM0, UARTM1) ... 279
UART Status Register (UARTS0, UARTS1) ... 283
UART Transmit Buffer (TxB0, TxB1) ... 286

[W]
Watchdog Timer Mode Register (WDM) ... 389

[MEMO]

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-889-1689

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	❑	❑	❑	❑
Technical Accuracy	❑	❑	❑	❑
Organization	❑	❑	❑	❑